Die semantische Gliederung zur Repräsentation des Bedeutungsinhalts innerhalb sprachverstehender Systeme

Johannes Müller

Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Die semantische Gliederung zur Repräsentation des Bedeutungsinhalts innerhalb sprachverstehender Systeme

Johannes Müller

Vollständiger Abdruck der von der Fakultät
für Elektrotechnik und Informationstechnik
der Technischen Universität München
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs
genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. techn. J. Swoboda
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. rer. nat. M. Lang
	2.	Univ.-Prof. Dr.-Ing. G. Färber

Die Dissertation wurde am 16.01.1997 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 10.06.1997 angenommen.

Zusammenfassung

In der vorliegenden Arbeit wird die *semantische Gliederung* als eine neuartige Repräsentation des Bedeutungsinhaltes einer Äußerung innerhalb eines sprachverstehenden Systems vorgestellt. Da sie eine probabilistische Aussage über die ihr zugrundeliegende Wortkette erlaubt, wird die unmittelbare Konvertierung einer Beobachtungsfolge oder Merkmalsvektorenfolge eines Sprachsignals in eine solche semantische Gliederung mittels eines rein stochastischen ‚top-down'-Ansatzes ermöglicht.

Als Beispielapplikation wurde ein *sprachverstehender Grafikeditor* implementiert, mit dem ein Benutzer dreidimensionale Objekte auf dem Bildschirm mittels Kommandierung in natürlicher, gesprochener Sprache erzeugen, verändern oder löschen kann. Durch Übertragung der Algorithmen in einen *sprachverstehenden Serviceroboter* konnte der anschauliche Nachweis der System-Portabilität erbracht werden. Innerhalb beider Domänen ist die semantische Gliederung in mehrfacher Hinsicht von zentraler Bedeutung:

- Sie stellt das Ergebnis einer rein probabilistischen *semantischen Decodierung* von einer gesprochenen oder geschriebenen Äußerung dar.
- Sie ist die Eingabeschnittstelle für den *Intentionsdecoder* zur sprachlichen Steuerung einer laufenden Applikation. Der von diesem Modul gelieferte maschineninterpretierbare Code, der an die Applikation weitergereicht wird, wird als die der Äußerung zugrundeliegende Intention betrachtet. Im Unterschied zur semantischen Gliederung beinhaltet die Intention jedoch die aktuelle Umgebungskonstellation.
- Sie dient als Eingabeschnittstelle für die semantikbasierte *Sprachproduktion* zur Generierung einer natürlichsprachlichen Wortkette. Eine Kombination aus semantischer Decodierung und Sprachproduktion ermöglicht einen einfachen Aufbau eines Systems zur semantikbasierten *automatischen Übersetzung* von natürlicher, gesprochener oder geschriebener Sprache in eine andere.

Um die zur semantischen Decodierung verwendeten stochastischen Wissensbasen (semantisches, syntaktisches, phonetisches, akustisches Modell) zu trainieren, d.h. deren Wahrscheinlichkeiten zu bestimmen, mußte zunächst geeignetes Sprachmaterial zur Verfügung gestellt werden. Im Rahmen einer durchgeführten *‚Wizard-of-Oz'-Simulation*, bei der ein menschlicher ‚Wizard' (‚Zauberer') die zu trainierende, sprachverstehende Komponente des Rechners ohne Wissen des Benutzers übernahm, konnten nahezu 2000 authentische, gesprochene Äußerungen aus der Grafikeditor-Domäne gesammelt werden. Dabei ließ sich eine deutliche Korrelation zwischen der technologischen Vorbildung eines Sprechers und dem verwendeten Satzbau feststellen.

Ein großes Problem für sprachverarbeitende Systeme stellen im Vokabular nicht enthaltene Worte dar (‚Out-of-Vocabulary': OOV). Im Rahmen dieser Arbeit wurde ein quantitatives Verfahren entwickelt, mit dem aus bestehenden Trainingsdaten auf die Wahrscheinlichkeit für das Auftreten mindestens eines unbekannten Wortes, der *OOV-Rate*, geschlossen werden kann. Desweiteren läßt sich damit die erforderliche Trainingskorpusaufstockung zur Verringerung der OOV-Rate unter einen bestimmten Wert abschätzen.

Sicherlich ist die benutzerseitige *Akzeptanz* von Bedeutung, wenn der sinnvolle Einsatz einer Applikation diskutiert wird. Gerade bei einer neuen Technologie wie der Sprachverarbeitung muß gezielt auf eine breite Akzeptanz geachtet werden. Einige Untersuchungen zu diesem Thema wurden durchgeführt und ergänzen diese Arbeit.

Vorwort

Die vorliegende Arbeit entstand während meiner Zeit als wissenschaftlicher Assistent am Lehrstuhl für Mensch-Maschine-Kommunikation der Technischen Universität München.

An erster Stelle möchte ich mich bei meinem Doktorvater Univ.-Prof. Dr. rer. nat. Manfred Lang für die stete Betreuung dieser Arbeit und für sein offenes Ohr für auftauchende Probleme bedanken. Die von ihm ermöglichte freie Arbeitsweise erleichterte mir, wissenschaftliches Neuland zu betreten und unkonventionelle Ideen umzusetzen, was auf die vorliegende Arbeit einen sehr fruchtbaren Einfluß hatte.

Bei den Herren Dr. Harald Höge, Dr. Erwin Marschall und Gerhard Niedermair von der SIEMENS-Sprachverarbeitungsgruppe bedanke ich mich für die intensive Nachhilfe auf den Gebieten der statistischen Spracherkennung und der linguistischen Analyse.

Auf Anregung von Univ.-Prof. Dr.-Ing. Günther Schmidt, Ordinarius am Lehrstuhl für Steuerungs- und Regelungstechnik, konnte ein interdisziplinäres und lehrstuhlübergreifendes Projekt beginnen. In diesem Zusammenhang danke ich seinem wissenschaftlichen Mitarbeiter Christian Fischer und dessen Diplomanden Peter Havel für die sehr interessante und überaus erfolgreiche Zusammenarbeit.

Natürlich gebührt mein Dank den Diplomanden, Studienarbeitern und Werkstudenten, die innerhalb des Forschungsgebietes „Natürlichsprachlicher Mensch-Maschine-Dialog“ am Lehrstuhl tätig waren und der vorliegenden Arbeit brauchbare Mosaiksteine beisteuerten. Besonders danken möchte ich Michael Ebersberger, der sein intensives wissenschaftliches Engagement selbst noch nach seiner Diplomandenzeit fortsetzte.

Für die tatkräftige Unterstützung in allen Bereichen der Hard- und Software und für die prompte Erledigung aller diesbezüglichen Probleme möchte ich den Systemadministratoren Peter Brand und Heiner Hundhammer vielmals danken.

Was wäre ein Arbeitstag ohne Frau Christine Reischer? Ihre tatkräftige, organisatorische Unterstützung und ihre ehrliche Meinung erleichterten meine Zeit am Lehrstuhl erheblich. Vielen, vielen Dank!

Mein Dank gebührt auch meinen Kollegen Peter Morguet, Hans-Jürgen Winkler und Robert Zwickenpflug aus der Arbeitsgruppe Mensch-Maschine-Kommunikation für ihre spontane Diskussionsbereitschaft und die stets kollegiale Zusammenarbeit.

Das abschließende und besonders große Dankeschön gilt meinem Kollegen und Freund Holger Stahl für die unzähligen Ideen und Anregungen, die fruchtbaren Diskussionen sowie das ausgesprochen gute Arbeitsklima. Die von uns erzielte Synergie wirkte sich enorm kreativ und positiv auf die vorliegende Arbeit aus.

Holzkirchen und München im Januar 1997

Johannes Müller

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Sprachverarbeitung zur Mensch-Maschine-Kommunikation

Spracherkennung, *Sprachverstehen* und *Sprachübersetzung* sind drei Disziplinen der automatischen Sprachverarbeitung, welche für eine benutzeradäquate Mensch-Maschine-Kommunikation [Lan94c][Lan94d] sehr wünschenswert wären, jedoch technisch und technologisch derzeit noch nicht hinreichend beherrscht werden. Natürliche Sprache[1)] ist das meistverwendete Kommunikationsmittel zwischen den Menschen, wird in der Regel von jedem beherrscht, funktioniert ohne den Einsatz von Händen oder Füßen und ist ohne visuelle Rückkopplung und damit auch im Dunklen möglich [Lan86][Lan91][Lan94b]. Dabei soll der Rechner nicht in die Lage versetzt werden, natürliche Sprache wie ein Mensch kognitiv zu erfassen und damit Emotionen oder Assoziationen zu wecken, sondern lediglich Sprache als Informationsquelle zur Erledigung vorgegebener Aufgaben verwenden. In diesem Sinne seien drei Anwendungsgebiete exemplarisch dargestellt, die mit den drei obigen Disziplinen ausgeführt werden können.

- Spracherkennung: „Hörende Schreibmaschine“
- Sprachverstehen: Natürlichsprachlicher Datenbankzugriff
- Sprachübersetzung: Übersetzungssystem einer natürlichen Sprache in eine andere

Spracherkennung ist die alleinige Umwandlung von gesprochener Sprache in geschriebenen Text, das kann ein einzelnes Wort *w* oder eine aus mehreren Worten bestehende Wortkette *W* sein. Ein mögliches Anwendungsbeispiel ist die „hörende Schreibmaschine“, bei welcher der Computer ein gesprochenes Diktat als graphemische Wortkette *W* niederschreiben, jedoch nicht interpretieren muß.

---

1) Das Wort „Sprache“ ist im Deutschen mehrdeutig und kann sich entweder (im Sinne von *speech*) auf gesprochene Sprache oder (im Sinne von *language*) auf geschriebene Sprache beziehen. Falls nicht anderweitig angegeben, bezieht sich in dieser Arbeit *Sprache* immer auf *natürliche, gesprochene Sprache*. Für *natürliche, geschriebene Sprache* wird im weiteren der Ausdruck *Text* verwendet. Diese Unterscheidung gilt analog für Ausdrücke wie Sprach-/Textverstehen oder Sprach-/Textübersetzung.

$$\text{Spracherkennung: Sprache} \rightarrow W \tag{1.1}$$

Hierbei ist der Bedeutungsinhalt des gesprochenen oder geschriebenen Textes völlig irrelevant für die Reaktion des Programms. Ein Einzelworterkenner (z.B. wie in [Gra95] beschrieben und implementiert) kann nur für sich losgelöste Worte erkennen. Dies kann auch insofern erweitert sein, daß beim Diktieren eines zusammenhängenden Textes kurze Pausen zwischen den einzelnen Wörtern gemacht werden müssen[2)]. Anspruchsvoller sind Systeme zur Erkennung fließender Sprache. Dabei sind zwischen den Worten keine Pausen notwendig, wobei jedoch das Problem der Koartikulation[3)] zu bewältigen ist. *Sprecherabhängige* Systeme sind nur auf die Ausspracheeigenheiten eines bestimmten Sprechers angepaßt, während *sprecherunabhängige* Systeme gesprochene Eingaben von beliebigen Sprechern verarbeiten können.

Sprachverstehen bedeutet allgemein die Interpretation von Sprache, d.h. die Extraktion der zugrundeliegenden Benutzerintention *I*. Im Falle der Abfrage oder Manipulation einer Datenbank wäre die betreffende Intention die gewünschte Umwandlung der sprachlichen Eingabe in computerverständliche Anweisungen, die den vom Benutzer sprachlich angeordneten Datenbankzugriff ausführen:

$$\text{Sprachverstehen: Sprache} \rightarrow I \tag{1.2}$$

Die obige Umwandlung kann dabei unmittelbar oder indirekt über eine oder mehrere Zwischenebenen (z.B. Wortebene, Semantikebene) vor sich gehen. Auch hier existieren die unterschiedlichen Disziplinen *Verstehen einzelner Worte* versus *Verstehen fließender Sprache*, desweiteren kann das Verstehen gesprochener Sprache sprecherabhängig oder sprecherunabhängig realisiert sein.

Sprachübersetzung ist die anspruchsvollste Disziplin und stellt die automatische Übersetzung einer natürlichen Sprache in eine andere dar. Da hierbei jedoch die Probleme der Sprachsynthese außer acht gelassen werden sollen, sei Sprachübersetzung als Umsetzung von gesprochener Quellsprache in geschriebene Zielsprache (Wortkette W_{ziel}) definiert:

$$\text{Sprachübersetzung: Sprache} \rightarrow W_{\text{ziel}} \tag{1.3}$$

Analog zum Sprachverstehen kann die Umwandlung (1.3) unmittelbar oder indirekt über eine oder mehrere Zwischenebenen (z.B. Wortebene, Interlingua-Ebene) vor sich gehen. Ebenfalls lassen sich *Übersetzung einzelner Worte* versus *Übersetzung fließender Sprache*, sowie sprecherabhängige versus sprecherunabhängige Übersetzung unterscheiden.

In dieser Arbeit wird innerhalb eines sprecherunabhängigen Systems zum Verstehen und Übersetzen fließender Sprache die sogenannte *semantische Gliederung* als syntaktisch-semantische Repräsentation einer gesprochenen Äußerung entwickelt, aus welcher eine Benutzerintention *I* oder eine Zielsprachen-Wortkette W_{ziel} abgeleitet werden kann. In

2) Zum Beispiel Diktiersystem *Tangora* der Firma IBM.

3) Phonetische Verschmelzung zweier benachbarter Worte bei der Aussprache.

letzterem Fall fungiert die semantische Gliederung als Interlingua-Ebene zwischen der Quellsprache und der Zielsprache.

1.2 Repräsentationsebenen einer sprachlichen Äußerung

Wie bereits in vorigem Kapitel angedeutet, kann eine sprachliche Äußerung auf verschiedenen Ebenen repräsentiert sein. Allgemein kann der Verarbeitungsprozeß natürlicher Sprache aufgefaßt werden als Zusammenwirken dieser Hierarchieebenen, deren Verknüpfung entweder regelbasiert oder statistisch hergestellt wird [Hög93][Lan84]. Analog zur abstrakten menschlichen Verarbeitung spricht man von einer „unteren“ oder „niedrigeren“ Ebene, falls es sich um eine sprachsignalnahe Ebene handelt, und von einer „oberen“ oder „höheren“ Ebene, falls es sich um eine intentionsnahe Ebene handelt. Die Umwandlung einer vorgegebenen Äußerung von einer Ebene in eine andere kann signalgesteuert von „unten“ nach „oben“ (*bottom-up*) oder erwartungsgesteuert von „oben“ nach „unten“ (*top-down*) verlaufen. Folgende Ebene können im allgemeinen unterschieden werden:

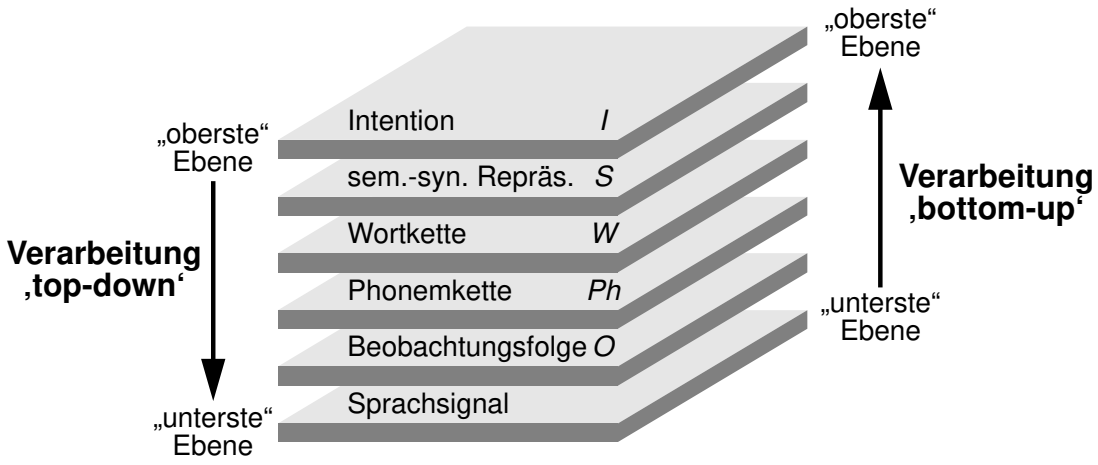


Abb. 1.1: Repräsentationsebenen einer sprachlichen Äußerung

- Das **Sprachsignal** gilt als die „unterste“ Ebene. Es liegt zur rechnergestützten Signalverarbeitung in der Regel digital und bandbegrenzt vor. Dazu wird das zunächst zeit- und wertkontinuierliche Signal mit einer bestimmten Abtastfrequenz abgetastet und damit zeitdiskretisiert. Die jeweiligen Amplituden werden innerhalb einer festgelegten Quantisierungsbreite (d.h. mit konstanter Bitanzahl) auf bestimmte Amplitudenstufen wertdiskretisiert. In dieser Arbeit werden die Einstellungen

  $$\text{Abtastfrequenz} = 16\ \text{kHz}$$
  $$\text{Quantisierungsbreite} = 16\ \text{bit}$$

  verwendet, was einer Datenrate von $256000\ \frac{\text{bit}}{\text{s}} = 32\ \frac{\text{kByte}}{\text{s}}$ entspricht. (Im Vergleich dazu hat der digitale Telefonstandard ISDN mit einer Abtastfrequenz von 8 kHz und einer Quantisierungsbreite von 8 bit eine Datenrate von $64000\ \frac{\text{bit}}{\text{s}} = 8\ \frac{\text{kByte}}{\text{s}}$.)

- Die **Beobachtungsfolge (oder Merkmalsvektorenfolge) *O*** wird durch Vorverarbeitung des Sprachsignals gewonnen [Lan94a]. Dazu werden sehr kurze, im vorliegenden Fall 25 ms dauernde Signalabschnitte[4)] aus dem Gesamtsignal extrahiert (d.h. „gefenstert“) und anschließend einer Fast-Fourier-Transformation (FFT) unterzogen. Um Instationaritäten des Sprachsignal gut erfassen zu können, wird diese Kurzzeitanalyse fortlaufend alle 10 ms durchgeführt. Aus dem geglätteten Kurzzeitspektrum werden nun mehrere Stützpunkte abgetastet, welche die einzelnen Komponenten eines Merkmalsvektors bilden.

  Indem die Wahl der spektralen Stützpunkte nicht äquidistant, sondern in den Abständen der sogenannten Mel-Skala [Zwi82] gewählt wird, kann die Frequenzauflösung des menschlichen Gehörs berücksichtigt werden. Eine eventuell nachfolgende Koordinatentransformation – beruhend auf der linearen oder nicht-linearen Diskriminanzanalyse [Hae92] – bewirkt einen maximalen Informationsgehalt zur Klassifizierung der Äußerung. Eine solche Signalvorverarbeitung erzeugt somit alle 10 ms einen mehrdimensionalen Merkmalsvektor $\overrightarrow{o_{no}}$, der die spektralen Eigenschaften des Sprachsignals beschreibt. Die sequentielle Aneinanderreihung solcher Merkmalsvektoren $\overrightarrow{o_{no}}$ nennt man *Merkmalsvektorenfolge* oder *Beobachtungsfolge* (Observation sequence) *O*. Die Anzahl der in *O* enthaltenen Merkmalsvektoren ist *NO*.

$$O = \overrightarrow{o_1}\ \overrightarrow{o_2}\ \ldots\ \overrightarrow{o_{no}}\ \ldots\ \overrightarrow{o_{NO}} \tag{1.4}$$

- Die **Phonemkette *Ph*** repräsentiert die phonetische Aussprache der gesprochenen Äußerung. Sie ist eine sequentielle Aneinanderreihung der ausgesprochenen Phoneme ph_{np}. Wortgrenzen sind in der Phonemkette nicht enthalten. Die Anzahl der in *Ph* enthaltenen Phoneme ist *NP*.

$$Ph = ph_1\ ph_2\ \ldots\ ph_{np}\ \ldots ph_{NP} \tag{1.5}$$

- Die **Wortkette *W*** ist eine durch grammatische und lexikalische Regeln festgelegte Repräsentation einer sprachlichen Äußerung. Sie ist jedermann ohne besonderes logisches oder formales Vorwissen bekannt, kann vom Menschen auch schnell gelesen und interpretiert werden und eignet sich deshalb besser als alle anderen Ebenen als Kontrollschnittstelle. In der Regel benötigt die Wortkette den geringsten Speicherplatz im Vergleich zu allen anderen hier betrachteten Ebenen. In dieser Arbeit sei als Wortkette lediglich die Abfolge von Worten ohne jegliche Satzzeichen verstanden. Sie ist damit eine sequentielle Aneinanderreihung der vorkommenden Worte w_{nw}. Die Anzahl der in *W* enthaltenen Worte ist *NW*.

$$W = w_1\ w_2\ \ldots\ w_{nw}\ \ldots\ w_{NW} \tag{1.6}$$

- Als **semantisch-syntaktische Repräsentationsebene *S*** wird in dieser Arbeit die semantische Gliederung (siehe dazu Kap. 3 und [Mül95a]) als formale Darstellung ein-

4) Für die Fensterdauer hat sich der Bereich 10...30 ms bewährt. Eine Fensterdauer kleiner als 10 ms hätte eine schlechtere spektrale Auflösung zur Folge, bei einer Fensterdauer größer als 30 ms wäre die angenommene Quasistationarität innerhalb des Fensters nicht gewährleistet.

geführt. Sie bildet bei der Verarbeitung der Wortkette zur Intention eine notwendige und zweckmäßige Zwischenebene. Ja nach Bedarf kann die semantisch-syntaktische Repräsentation eher wort- oder eher intentionsnah sein, was sich auf die Komplexität der semantischen Decodierung oder der Intentionsdecodierung auswirkt. Die semantische Gliederung ist eine Menge von N semantischen Untereinheiten s_n, welche jeweils einen kleinen Anteil des Bedeutungsinhaltes der Äußerung verkörpern.

$$S = \{s_1, s_2, \ldots, s_n, \ldots, s_N\} \tag{1.7}$$

- Die **Intention *I*** gilt als „oberste“ Ebene. Sie repräsentiert genau diejenigen applikationsspezifischen Kommandos, die am Rechner die von Benutzer gewünschte Aktion auslösen oder ihm die gewünschte Information ausgeben.

$$I = \text{Maschinenbefehl}_1, \text{Maschinenbefehl}_2, \ldots \tag{1.8}$$

1.2.1 ‚Bottom-up‘- versus ‚Top-down‘-Ansatz

Ein sprachverstehendes System findet zu einem vorliegenden Sprachsignal die der Benutzerintention entsprechenden Maschinenbefehle. Ein naheliegender Lösungsansatz ist dabei eine rein signalgesteuerte Abarbeitung dieser Ebenen in ‚bottom-up‘-Richtung: Zu einem gegebenen Sprachsignal wird die Beobachtungsfolge gebildet, dazu anschließend die Phonemkette gesucht, welche wiederum in eine Wortkette überführt wird. Diese wird in eine semantisch-syntaktische Repräsentation gewandelt, woraus die Intention (d.h. die abzuarbeitenden Maschinenbefehle) generiert werden. Dieser reine ‚bottom-up‘-Ansatz klingt zwar durchaus plausibel und nachvollziehbar, denn die Entscheidungen, die getroffen werden müssen, um von einer Ebene in die nächsthöhere zu gelangen, sind meistens eindeutig:

- Zu einem Sprachsignal existiert genau eine Beobachtungsfolge O.
- Zu einer Beobachtungsfolge O existieren mehrere mögliche Phonemketten Ph, deren bedingte Wahrscheinlichkeiten jedoch unterschiedlich sein können.
- Zu einer Phonemkette Ph existiert zumeist genau eine Wortkette W.
- Zu einer Wortkette W existiert in der Regel genau eine semantische Gliederung S.
- Zu einer semantischen Gliederung S existiert in der Regel genau eine Intention I.

Der zunächst naheliegende ‚bottom-up‘-Ansatz ist jedoch mit enormen Nachteilen behaftet: Zu einer Transformation in die nächsthöhere Ebene kann lediglich Wissen aus diesen beiden Ebenen zur Verfügung stehen. Wird jedoch bei einer tieferliegenden Transformation eine Fehlentscheidung getroffen, was in erster Linie bei der extrem mehrdeutigen und unscharfen Umwandlung $O \rightarrow Ph$ zu erwarten ist, so pflanzt sich dieser Fehler in der Regel bis in die oberen Instanzen fort und kann dort nicht mehr behoben werden.

Das Gegenteil ist das rein erwartungsgesteuerte Vorgehen in ‚top-down‘-Richtung: Es werden viele mögliche Intentionen generiert. Zu jeder Intention werden korrespondierende semantische Gliederungen erzeugt. Jede davon produziert passende Wortketten,

jede Wortkette entsprechende Phonemketten und jede Phonemkette entsprechende Beobachtungsfolgen. Paßt nun das vorliegende Sprachsignal auf eine dieser Beobachtungsfolgen, muß die Entstehung dieser Beobachtungsfolge „nach oben“ zurückverfolgt werden, um die zugehörige Intention *I* zu erhalten. Das Problem beim erwartungsgesteuerten Ansatz ist sicherlich die große Mehrdeutigkeit bei der Umwandlung einer „höheren“ Ebene in eine „tiefere“:

- Es existieren prinzipiell unendlich viele denkbare Intentionen *I*.
- Zu einer Intention *I* können viele verschiedene syntaktisch-semantische Darstellungen *S* existieren.
- Zu einer syntaktisch-semantischen Darstellung *S* können viele verschiedene Wortketten *W* existieren.
- Zu einer Wortkette *W* können mehrere verschiedene Phonemketten *Ph* existieren.
- Zu einer Phonemkette *Ph* können extrem viele verschiedene Beobachtungsfolgen *O* existieren.
- Zu einer Beobachtungsfolge *O* können unendlich viele verschiedene Sprachsignale existieren.

Ein solches Vorgehen ist jedoch nicht praktikabel – der zur Verfügung stehende Speicherplatz würde angesichts der eigentlich unendlich vielen Hypothesen nicht ausreichen. Dieses Vorgehen muß stückweise über der Zeit erfolgen: Zu einem Zeitpunkt unzutreffende oder geringwahrscheinliche Hypothesen werden verworfen und zu späteren Zeitpunkten nicht mehr betrachtet[5)]. Die extreme Mehrdeutigkeit hat zwar eine extrem rechen- und speicherplatzaufwendige Implementierung zur Folge. Trotzdem hat sich der ‚top-down‘-Ansatz in der Spracherkennung bestens bewährt.

1.2.2 Regelbasierter versus statistischer Ansatz

Ein regelbasierter Algorithmus ist im Grunde ein Expertensystem, welches unter bestimmten Vorbedingungen festgelegte Entscheidungen trifft. Wenn also eine ganz bestimmte Phonemkette vorliegt, dann kann in ‚bottom-up‘-Richtung eindeutig auf die zugehörige Wortkette geschlossen werden. Wie in diesem Satz angedeutet, kann dies mit einem Inferenzmechanismus („wenn...dann“-Regeln) geschehen. Bei eindeutigen Übergängen sind statistische Bindungen unnötig, da die relevanten Wahrscheinlichkeiten nur einen der Werte Null oder Eins annehmen können. Prinzipiell kann ein regelbasierter Ansatz auch durch einen statistischen Ansatz ausgedrückt werden: „Erlaubte“ Beziehungen zwischen zwei Ebenen hätten die Wahrscheinlichkeit Eins, nicht vorkommenden Beziehungen würde die Wahrscheinlichkeit Null zugeordnet.

Sobald der Übergang von einer Ebene zu einer anderen mehrdeutig ist, ist eine wahrscheinlichkeitsbehaftete, d.h. probabilistische Verarbeitung sinnvoll. Ein solcher statistischer Ansatz ermöglicht mehrere Entscheidungen, die allerdings mit unterschiedlichen

5) Dieses Vorgehen wird in der Sprachverarbeitung als „Pruning“ (auf Deutsch „Zusammenstreichung, Kürzung) bezeichnet.

Wahrscheinlichkeiten behaftet sind. Zum Beispiel können in ‚bottom-up'-Richtung zu einer gegebenen Beobachtungsfolge O mehrere Phonemketten korrespondieren. Die Wahrscheinlichkeit des Auftretens einer bestimmten Phonemkette Ph_i wäre $P(Ph_i|O)$. Die Summe der Wahrscheinlichkeiten aller Phonemketten Ph_i, die zu einer Beobachtungsfolge O gebildet werden können, ist erwartungsgemäß gleich Eins:

$$\sum_{\text{alle } i} P\left(Ph_i|O\right) \;=\; 1 \tag{1.9}$$

In ‚top-down'-Richtung wäre die Wahrscheinlichkeit $P(W|S)$ ein Maß für das Auftreten einer Wortkette W, falls die semantische Gliederung S vorliegt. Die Festlegung aller benötigten Wahrscheinlichkeiten ist die Aufgabe des *Trainings*, worauf in Kapitel 5 näher eingegangen wird.

Bei einer sprachverstehenden Applikation ist letztendlich diejenige Intention I_E gesucht, welche die Wahrscheinlichkeit $P(I|\text{Sprachsignal})$ maximiert:

$$I_E \;=\; \underset{I}{\text{argmax}}\; P(I|\text{Sprachsignal}) \tag{1.10}$$

Die Dimensionalität der Wahrscheinlichkeit $P(I|\text{Sprachsignal})$ wäre viel zu hoch, um Gl. (1.10) direkt zu maximieren. So erscheint es sinnvoll, die vorher definierten Ebenen in obige Gleichung mitaufzunehmen.

$$P(I|\text{Sprachsignal}) \;=\; \sum_{\text{alle } S} \sum_{\text{alle } O} \Big(P(I|S) \cdot P(S|O) \cdot P(O|\text{Sprachsignal}) \Big) \tag{1.11}$$

Eingesetzt in Gl. (1.10) ergibt dies:

$$I_E \;=\; \underset{I}{\text{argmax}} \sum_{\text{alle } S} \sum_{\text{alle } O} \Big(P(I|S) \cdot P(S|O) \cdot P(O|\text{Sprachsignal}) \Big) \tag{1.12}$$

Da die Umwandlung des Sprachsignals in eine Beobachtungsfolge O und die Überführung der semantischen Gliederung S in die Intention I nach festgelegten Regeln eindeutig verläuft, können diese Schritte zweckmäßigerweise regelbasiert erfolgen. Somit müßten bei Gl. (1.12) anstatt aller S und O nur das zum gesuchten I_E korrespondierende S_E und die zum Sprachsignal gehörende Beobachtungsfolge O betrachtet werden. Nur die semantische Decodierung mit dem Term $P(S|O)$ wird statistisch betrachtet. Aus diesen Vorüberlegungen ergeben sich zum Prozeß des Sprachverstehens drei sequentielle Verarbeitungsschritte.

- Die **Vorverarbeitung** erledigt die regelbasierte, algorithmische Transformation $\mathbf{T}_O$ des gegebenen Sprachsignals in eine korrespondierende Beobachtungsfolge oder Merkmalsvektorenfolge O.

$$O \;=\; \mathbf{T}_O\{\text{Sprachsignal}\} \tag{1.13}$$

- Der **semantische Decoder** sucht mit stochastischen Methoden diejenige semantische Gliederung S_E, welche die bedingte Wahrscheinlichkeit für das Auftreten einer semantischen Gliederung gegeben eine Beobachtungsfolge O maximiert.

$$S_E = \underset{S}{\operatorname{argmax}}\, P(S|O) \tag{1.14}$$

- Der **Intentionsdecoder** führt die regelbasierte Transformation $\mathbf{T}_I$ der semantischen Gliederung S_E in die für die Benutzerintention repräsentativen Maschinenbefehle aus. Eine Beschreibung dieses Moduls wird im Kapitel 6 gegeben.

$$I_E = \mathrm{T}_I\{S_E\} \tag{1.15}$$

Die Zusammenfassung der Module Vorverarbeitung, semantische Decodierung und Intentionsdecodierung wird im folgenden als *Sprachverstehen* bezeichnet. Aus den Gleichungen (1.13), (1.14) und (1.15) ergibt sich für das in dieser Arbeit praktizierte Sprachverstehen folgende, grundlegende Formel:

$$I_E = \mathrm{T}_I\left\{\underset{S}{\operatorname{argmax}}\, P(S \mid \mathrm{T}_O\{\mathit{S}\text{prachsignal}\})\right\} \tag{1.16}$$

Folgende Abbildung stellt die gewählte Vermischung der beschriebenen Möglichkeiten dar. Im Prinzip wird für das gesamte sprachverstehende System eine ‚bottom-up'-Richtung gewählt. Innerhalb dieses Ansatzes wird die Umwandlung einer Beobachtungsfolge O in eine semantische Gliederung S – diese Umwandlung stellt jedoch den wesentlichen Teil dar – nach statistischen Methoden in ‚top-down'-Richtung durchgeführt.

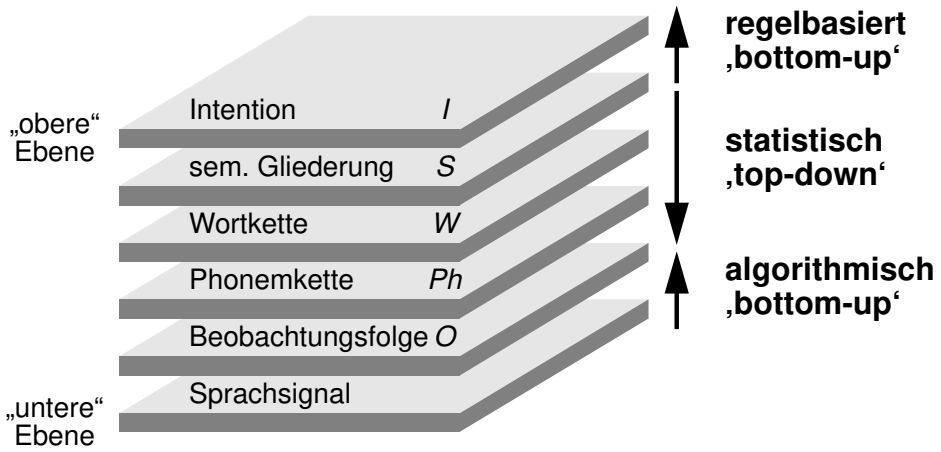


Abb. 1.2: Gewählte Aufteilung der Verarbeitungsschritte

Das in obiger Abbildung 1.2 gewählte Verarbeitungsprinzip läßt sich mit dem bewährten Ansatz von Pieraccini vergleichen. Er vergleicht Sprachverstehen mit einem Übersetzungsprozeß einer natürlichen Sprache *N-L* in eine Computersprache *C-L*, was prinzipiell auf zwei Stufen wie im folgenden Bild 1.3 dargestellt abläuft.

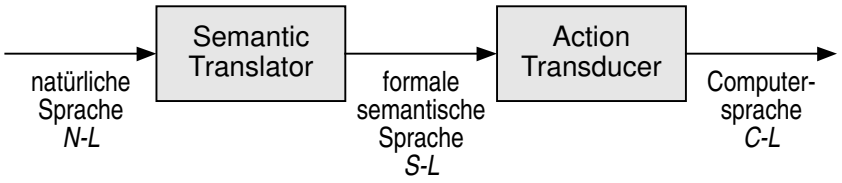


Abb. 1.3: Umwandlung von natürlicher Sprache in Computersprache (nach Pieraccini [Pie92c][Pie93])

Ein *Semantik Translator* (entspricht dem semantischen Decoder) übersetzt die natürliche Sprache *N-L*, die gesprochen oder geschrieben vorliegt, in eine formale semantische Sprache *S-L*. Daraus kann ein nachfolgender *Action Transducer* (entspricht dem Intentionsdecoder) eine direkt vom Computer verständliche Sprache *C-L* bilden. Die Komplexität dieser Module hängt wesentlich von der Abstraktionsebene von *S-L* ab. Der semantische Decoder kann für eine *N-L*-nahe formale semantische Sprache *S-L* relativ einfach gehalten werde. Der Aufwand steckt dann allerdings im Intentionsdecoder. Umgekehrt muß der semantische Decoder komplex und der Intentionsdecoder einfach sein, falls sich die semantische Sprache *S-L* auf einer hohen Abstraktionsebene befindet, d.h. *N-L*-fern und *C-L*-nah ist [Pie92c][Pie93].

1.3 Einschränkung auf begrenzte Domäne

Angesichts der unbeschränkten Vielzahl von zu erkennenden Worten und Wortketten ist es sinnvoll, die Domäne (d.h. das Gebiet, in dem sich eine zu erwartende Äußerung befindet) eng zu umgrenzen.

Es wurde ein System zum Verstehen gesprochener Äußerungen am Beispiel eines natürlichsprachlichen „Grafikeditors“ entworfen und vollständig implementiert. Mit dieser Applikation ist es möglich, mit gesprochenen Anweisungen eine dreidimensionale Anordnung aus Kegeln, Kugeln, Quadern und Zylindern zu editieren, d.h. einzelne Objekte zu erzeugen, zu verändern oder zu löschen. Trotz der etwas umständlichen Koordinateneingabe[6)] mittels natürlicher Sprache wurde diese Domäne gewählt, da sie für den Laien eine anschauliche Anwendung des Sprachverstehens darstellt, für die keinerlei Einarbeitung oder Lernphase notwendig erscheint.

Natürlich sind die in den folgenden Kapiteln beschriebenen Verfahren nicht nur zur Bedienung eines Grafikeditors verwendbar. Der Nachweis zur Portabilität der erarbeiteten Formalismen und Algorithmen sowohl auf andere Applikationen als auch auf andere Domänen wird in Kapitel 10 erbracht. Im Rahmen eines interdisziplinären Projektes wurde die Verwendung des sprachverstehenden Systems als natürlichsprachliche Mensch-Maschine-Schnittstelle für einen autonomen Service-Roboter [Dax96][Fis96a] untersucht und implementiert.

6) Optimal wäre hierbei eine multimodale Eingabe mittels Sprechen und Zeigen („Touch-Screen“). Die vorliegende Arbeit befaßt sich jedoch ausschließlich mit sprachlicher Eingabe.

1.4 Projekt „Speech to Graphics“

Das in der folgenden Abbildung 1.4 dargestellte Projekt wurde als erster Prototyp eines sprachverstehenden Systems realisiert. Während in einer zeitlich parallel laufenden Arbeit die im folgenden Bild links der Trennlinie dargestellten Module ‚Vorverarbeitung‘ und ‚Wortkettendecoder‘ bearbeitet wurden, sind die anderen, rechts der Trennlinie dargestellten Module im Rahmen der vorliegenden Arbeit implementiert. Als Schnittstelle fungiert dabei die vom Wortkettendecoder ausgegebene ‚first-best‘ Wortkette W.

Mit Hilfe des Wortkettendecoders, welcher Algorithmen des SPICOS-Systems [Hög90] beinhaltet, wurde ein mit natürlicher, gesprochener Sprache bedienbarer Grafikeditor aufgebaut. Die semantische Decodierung setzt sich hierbei aus zwei Stufen zusammen, bestehend aus Wortkettendecoder und semantischer Textanalyse. Als Schnittstelle zwischen diesen Modulen fungierte die beste Wortkettenhypothese W_E mit

$$W_E = \underset{W}{\operatorname{argmax}}\, P(W|O)\,, \tag{1.17}$$

d.h. diejenige Wortkette mit der höchsten Wahrscheinlichkeit, falls eine Beobachtungsfolge O vorliegt.

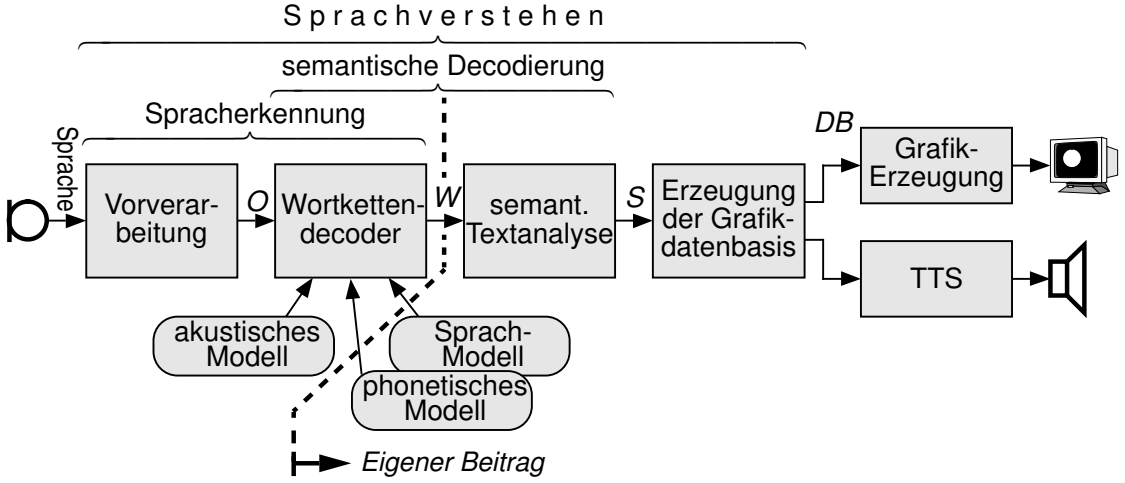


Abb. 1.4: Schematischer Aufbau des Projekts „Speech to Graphics“

Der Wortkettendecoder arbeitet sprecherunabhängig, erlaubt als Eingabe fließende Sprache und ermittelt die wahrscheinlichste Wortkette aufgrund einer Viterbi-Suche [Vit73] mit phonembasierten Hidden-Markov-Modellen (HMMen) [Hua90]. Die nachfolgende semantische Textanalyse arbeitet regelbasiert als ein unidirektionaler (links-rechts) Parsingalgorithmus, welcher aufgrund der syntaktischer Konstellation auf vorherrschende semantische Konzepte schließt. Aus Übersichtsgründen wird im weiteren auf eine exakte Darstellung dieses Ansatzes verzichtet.

Der Wortschatz und die Komplexität der Äußerungen waren jedoch bei diesem System sehr stark eingeschränkt. Auch die semantische Mächtigkeit war zu gering, so daß viele Benutzerintentionen nicht abgearbeitet werden konnten. Außerdem erwies sich die Zwei-

teilung der semantischen Decodierung als ziemlich unbefriedigend, da eine Fehlentscheidung des Wortkettendecoders in der Regel von der semantischen Textanalyse nicht mehr korrigiert werden konnte. Daher wurde ein weiteres Grafikeditorprojekt mit dem Ziel entwickelt, die semantische Decodierung als rein probabilistischen, einstufigen Prozeß mit Methoden, die sich im Bereich der Spracherkennung bewährt hatten, ablaufen zu lassen.

1.5 Projekt „NASGRA“

„NASGRA“ bedeutet **NA**türlich**S**prachlicher **GRA**fikeditor. Dieses in den folgenden Bildern dargestellte Projekt ist das hauptsächliche Ergebnis zweier zeitlich parallel laufender Dissertationen am Lehrstuhl für Mensch-Maschine-Kommunikation. Während die signalnahe Verarbeitung (in den folgenden Bildern links der jeweiligen Trennlinie dargestellt) von H. Stahl bearbeitet, implementiert und in dessen Dissertation [Sta97b] beschrieben wurde, stellt die intentionsnahe Verarbeitung (rechts der jeweiligen Trennlinie dargestellt) das Ergebnis der vorliegenden Arbeit dar. Die Schnittstelle zwischen diesen Arbeiten ist die im Kapitel 3 definierte semantische Gliederung als speziell für diesen Ansatz eingeführte semantisch-syntaktische Repräsentation einer sprachlichen Äußerung. Die stochastische Grammatik, die durch das Zusammenwirken der probabilistischen Wissensbasen semantisches und syntaktisches Modell festgelegt wird, wird in Kapitel 4 erläutert. Das gesamte, implementierte System läßt sich anhand dreier im folgenden dargestellter Teilsysteme anschaulich darstellen:

1.5.1 System zum Sprach- bzw. Textverstehen

Die folgende Abb. 1.5 demonstriert den sprachverstehenden Grafikeditor als Blockdiagramm. Die Eingabe kann einerseits gesprochene Sprache, aber andererseits auch geschriebener Text (vorliegend als Wortkette *W*) sein. Kapitel 2 beschreibt den grundlegenden Ansatz, welcher einen rein stochastischen semantischen Decoder mit vier konsistent genutzen, probabilistischen Wissensbasen (semantisches, syntaktisches, phonetisches und akustisches Modell) ermöglicht. Der nachfolgende Intentionsdecoder, welcher die

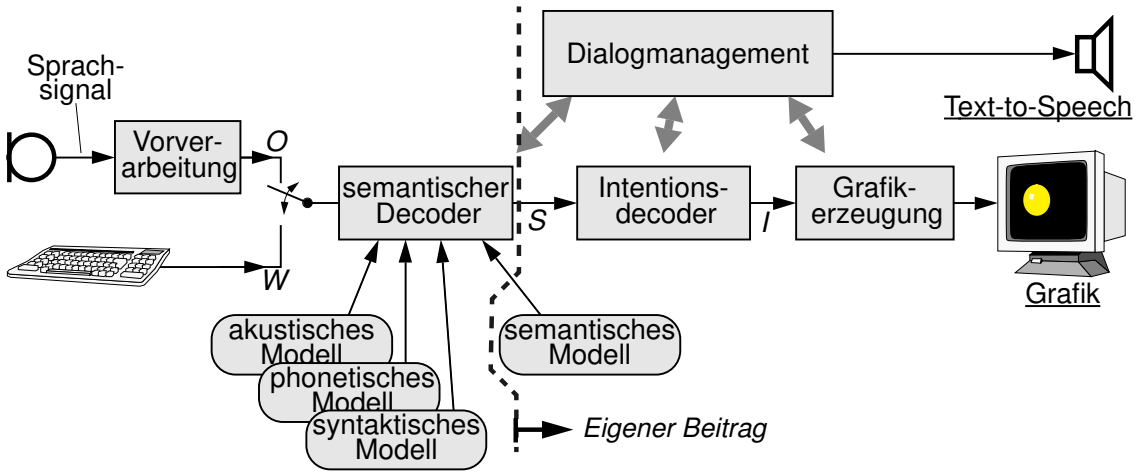


Abb. 1.5: Schematischer Aufbau der Beispielapplikation ‚sprachverstehender Grafikeditor‘

Intention *I* als maschineninterpretierbare Befehle erzeugt, wird im Kapitel 6 eingehend beschrieben. Richtlinien für einen zweckmäßigen, akustischen Mensch-Maschine-Dialog finden sich im Kapitel 7.

1.5.2 System zur Sprach- bzw. Textübersetzung

Abb. 1.6 zeigt den Einsatz in einem System zur semantikbasierten, automatischen Übersetzung domänenspezifischer Äußerungen, die im Kapitel 8 beschrieben ist. Aus einer gesprochenen oder geschriebenen Äußerung einer Quellsprache wird durch eine semantische Decodierung die als Interlingua-Ebene fungierende semantische Gliederung S_E ermittelt. Daraus produzieren die nachfolgenden Module ‚Wortkettengenerator' und ‚linguistische Nachbearbeitung' eine Wortkette W_{opt} in einer Zielsprache, die über Sprachsynthese akustisch ausgegeben werden kann. Im Rahmen dieser Arbeit wurden Wissensbasen für die Quellsprachen Deutsch und Slowenisch sowie für die Zielsprachen Deutsch, Englisch, Französisch und Slowenisch erstellt.

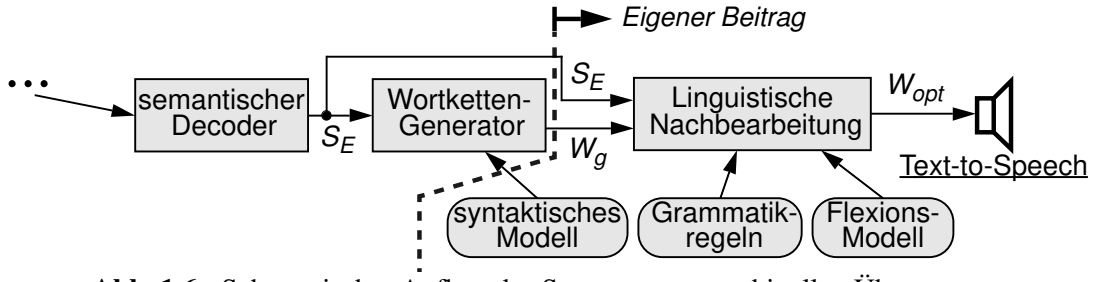


Abb. 1.6: Schematischer Aufbau des Systems zur maschinellen Übersetzung

1.5.3 Trainingsdatengewinnung, -verarbeitung und Training

Ein weiterer Bestandteil des Projektes war die im Kapitel 5 dargestellte Gewinnung von Trainingsmaterial, aus dem die zur semantischen Decodierung benötigten Wissensbasen generiert werden können. Für die Grafikeditor-Domäne wurden im Rahmen einer ‚Wizard of Oz'-Simulation von 33 verschiedenen Sprechern insgesamt 1843 gesprochene Äußerungen und durch eine Internet-Applikation ca. 2000 geschriebene Äußerungen gesammelt. Dazu kommen 285 Äußerungen für die Serviceroboter-Domäne. Desweiteren werden Verfahren zur quantitativen Bewertung der Trainingsmaterials untersucht und im Kapitel 9 beschrieben, so z.B. die Schätzung der Vokabulargröße und der ‚Out of Vocabulary'-Rate für die Erweiterung eines bestehenden Textkorpus'.

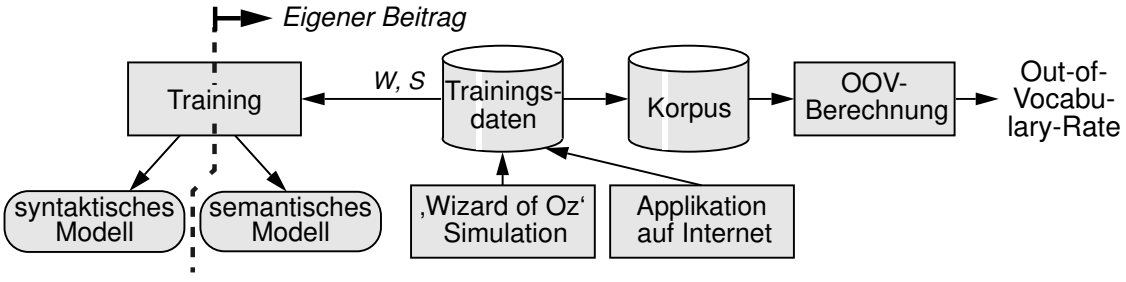


Abb. 1.7: Gewinnung und Weiterverarbeitung der Trainingsdaten, sowie Training der Modelle

Kapitel 2

Ansätze zur semantischen Decodierung gesprochener Sprache

Als semantische Decodierung wird in der vorliegenden Arbeit die Umwandlung einer Beobachtungsfolge in eine formale syntaktisch-semantische Darstellung verstanden. Bei einem einstufigen Ansatz erfolgt diese Umwandlung direkt, bei einem mehrstufigen Ansatz indirekt über eine oder mehrere Zwischenebenen (Phonem- bzw. Wortebene). Dabei bietet sich die Wortebene in besonderem Maße an, da dies eine jedermann vertraute Repräsentationsebene von natürlicher Sprache darstellt. Dieses Vorgehen hat einen zweistufigen Ansatz, bestehend aus Spracherkennung und semantischer Textanalyse, zur Folge.

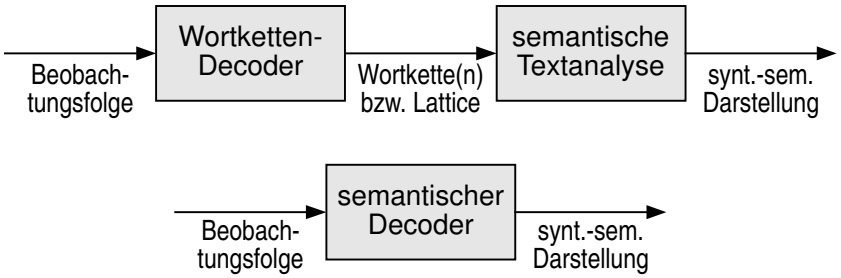


Abb. 2.1: Zwei- bzw. einstufiger Ansatz zur semantischen Decodierung von Sprache

2.1 Zwei- oder mehrstufiger Ansatz

Der derzeit meistverfolgte Ansatz zur semantischen Decodierung gesprochener Sprache ist eine Aufteilung dieses Prozesses in zwei voneinander getrennte Module. Die *Wortketten-Decodierung* liefert eine bestimmte Anzahl der n besten Wortketten-Hypothesen oder eine Wort-Lattice [Oer93], das ist ein Netzwerk aus Worthypothesen mit dazwischenliegenden Übergängen. Die nachfolgende *semantische Textanalyse* analysiert dies und generiert daraus eine korrespondierende syntaktisch-semantische Darstellung. Diese klassische Aufteilung hat ihren Ursprung durch die klare Trennung der eher ingenieurnahen Disziplin Signalverarbeitung (Spracherkennung) und der eher geisteswissenschaftsnahen

Disziplin Linguistik (semantische Textanalyse), welche mit der eindeutig definierten Schnittstelle der Wortebene miteinander verbunden sind.

2.1.1 Beispiele aus der aktuellen Forschung

- **CHRONUS-System (von AT&T Bell Laboratories, USA)**
  Ein gutes Beispiel für einen mehrstufigen Ansatz zur semantischen Decodierung ist das CHRONUS[7)]-System von AT&T Bell Laboratories, USA. Es stellt ein sprachverstehendes System zur Flugauskunft (Air Travel Information Service – ATIS [Pri90]) dar. Eine gesprochene Eingabe wird durch das spracherkennende Modul in eine Wort-Lattice gewandelt. Die daran anschließende semantische Analyse ist innerhalb dieses Systems nochmals in zwei Abschnitte unterteilt:

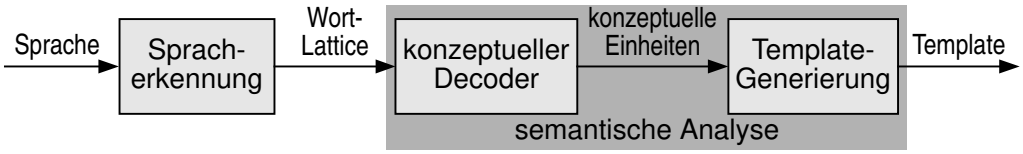


Abb. 2.2: Blockdiagramm der semantischen Decodierung von CHRONUS (übersetzt nach Pieraccini und Levin [Lev95][Pie95])

Ein konzeptueller Decoder wandelt die Wort-Lattice in *konzeptuelle Einheiten*, einer noch recht wortnahen semantischen Darstellung. Jedem Wort entspricht dabei eine bestimmte konzeptuelle Einheit, so daß Teilphrasen aus zusammenhängenden Worten bestimmte Konzepte (z.B. Flug-Start, Flug-Ziel, Stop-Anzahl) zugeordnet werden können. Diese Umwandlung geschieht mit stochastischen Methoden, einem sogenannten konzeptuellen HMM, welches in folgender Abbildung dargestellt ist:

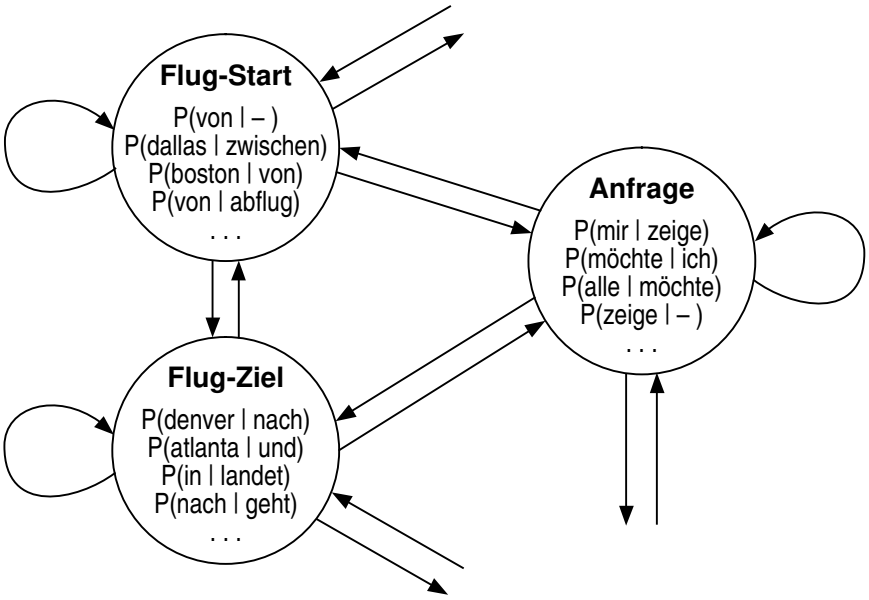


Abb. 2.3: Konzeptuelles HMM (übersetzt nach Pieraccini [Pie92b])

7) Conceptual Hidden Representation of Natural Unconstrained Speech

Die nachfolgende Template-Generierung erzeugt daraus mit einem regelbasierten Algorithmus sogenannte *Templates*. Sie sind eine abstrakte und sprachfernere Darstellung des semantischen Inhalts. Erst diese Darstellung ermöglicht eine sinnvolle Weiterverarbeitung in eine computerinterpretierbare Kommandos [Lev95][Pie91a] [Pie91b][Pie92a][Pie92b][Pie95][Tzo92].

Ein Beispiel für die Darstellung einer denkbaren Äußerung innerhalb der Domäne „Fluginformation und Flugbuchung“ ist in folgender Tabelle in deutsche Sprache übersetzt aufgeführt:

Wortkette:	ich möchte alle nonstop flüge von münchen nach frankfurt die am zweiten april starten	
konzeptuelle Einheiten:	Anfrage:	ich möchte alle
	Stop-Anzahl:	nonstop
	Anfrage-Objekt:	flüge
	Flug-Start:	von dallas
	Flug-Ziel:	nach denver
	Flug-Datum:	die am zweiten april starten
Template:	Anfrage:	ausgeben
	Objekt:	Gesellschaft+Nummer+Zeit
	Stops:	0
	Start-Flughafen:	MUC
	Ziel-Flughafen:	FRA
	Tag-Name:	Sonntag

Tab. 2.1: Darstellung von Äußerungen innerhalb CHRONUS
(übersetzt nach Pieraccini [Pie92a])

- **System zur Fluginformation und Flugbuchung (von IBM, Italien)**
  Einen dem CHRONUS-System sehr ähnlichen Ansatz innerhalb derselben Domäne beschreiten Bianchini et al. [Bia93] mit ihrem Fluginformations- und Flugbuchungssystem. Auch hier ist die semantische Analyse in zwei Module unterteilt: Die syntaktische Analyse wandelt den von Spracherkenner ausgegebenen erkannten Satz in einen sogenannten *syntaktischen Pfad*. Dieser wird von einer nachfolgenden semantischen Analyse in eine *Informationstabelle* umgeformt. Es liegt auf der Hand, daß hierbei prinzipiell der syntaktische Pfad den konzeptuellen Einheiten, die Informationstabelle den Templates ähnelt.

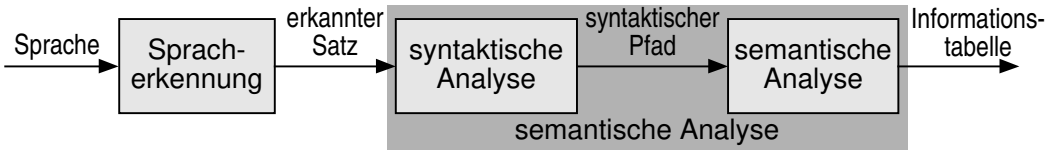


Abb. 2.4: Blockdiagramm Flugbuchungssystems von IBM, Italien
(nach Bianchini et al. [Bia93])

- **ATIS- und MASK-System (von LIMSI-CNRS, Frankreich)**
  Das ATIS-System von LIMSI-CNRS in Frankreich ist ein weiterer Vertreter innerhalb der Domäne Fluginformation. Eine weitere Domäne ist der Bereich Zugauskunft (MASK). Das gesamte sprachverstehende Konzept ist mit demjenigen von AT&T vergleichbar. Ein Modul zur Erkennung fließender, gesprochener Sprache mit einem Bi-

gramm-Sprachmodell und kontextabhängigen CDHMMen[8)] liefert eine Wortkette, die in einem nachfolgenden Analysemodul in sogenannte *semantische Frames*, einer speziellen Repräsentationsebene des Bedeutungsinhalts, umgewandelt wird [Lam95a] [Lam95b].

- **System für geographische Auskünfte in spanischer Sprache**
  Ein außergewöhnlicher und selten implementierter, zweistufiger Ansatz wurde von Prieto et al. innerhalb eines Systems für geographische Auskünfte in spanischer Sprache entwickelt [Pri94]. Im Gegensatz zu den bisherigen Beispielen fungiert hierbei als Zwischenebene nicht die Wortebene mit Worthypothesen, sondern die Phonemebene mit Phonemhypothesen.

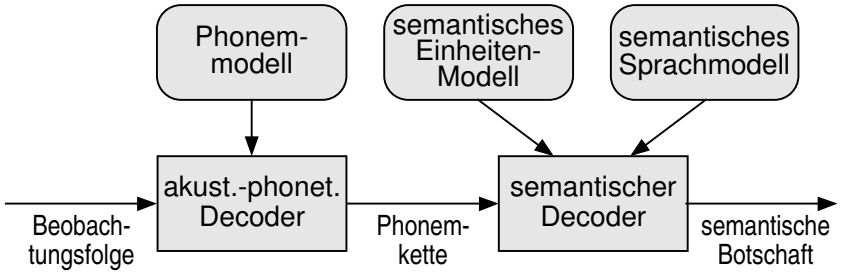


Abb. 2.5: Blockdiagramm einer zweistufigen, semantischen Decodierung mit der Phonemkette als Zwischenebene (nach Prieto et al. [Pri94])

Innerhalb dieses zweistufigen Ansatzes (*decoupled bottom-up architecture*) wird eine Beobachtungsfolge zunächst in eine Phonemkette gewandelt. Das dazu nötige Wissen wird aus einem sogenannten Phonemmodell bezogen. Anschließend findet mittels weiterer, semantischer Wissensbasen eine Umsetzung in die korrespondierende semantische Botschaft (*semantic message*) statt. Da bei der vorliegenden Architektur im ersten Verarbeitungsschritt lediglich akustisch-phonetisches Wissen zur Verfügung steht, kann die phonemische Zwischenebene bereits mit vielen Fehlern behaftet sein, so daß aus den bereits vorliegenden Wissensbasen ein im Kapitel 2.2.1 beschriebener einstufiger Ansatz entstand.

2.1.2 Wortkettendecodierung mit stochastischem Ansatz

Der Wortkettendecoder ermittelt aus einer vorliegenden Beobachtungsfolge O diejenige erkannte Wortkette W_E, welche unter Vorliegen genau dieser Beobachtungsfolge O und den zur Verfügung stehenden Wissensbasen die höchste Wahrscheinlichkeit besitzt:

$$W_E = \underset{W}{\operatorname{argmax}}\, P(W|O) \tag{2.1}$$

8) Ein CDHMM (Continuous Density Hidden-Markov-Model) ist ein Hidden-Markov-Model, dessen Emissionen kontinuierliche Wahrscheinlichkeitsdichteverteilungen und keine diskreten Ereignisse sind.

Diese Gleichung kann mit der Bayes'schen Regel umgeformt werden. Die sich ergebende a-priori-Wahrscheinlichkeit $P(O)$ muß für die Maximierung nicht berücksichtigt werden, da sie bei gegebener Beobachtungsfolge O konstant ist.

$$W_E = \operatorname*{argmax}_W \frac{P(O|W) \cdot P(W)}{P(O)} = \operatorname*{argmax}_W \left[P(O|W) \cdot P(W)\right] \tag{2.2}$$

Die direkte Bestimmung von $P(O|W)$ ist aufgrund der Vielfalt möglicher Kombinationen aus O und W nicht ohne weiteres möglich. Deshalb wird die Phonemebene Ph als weitere Repräsentationsebene eingeführt.

$$W_E = \operatorname*{argmax}_W \sum_{\text{alle } Ph} \left[P(O|Ph) \cdot P(Ph|W) \cdot P(W)\right] \tag{2.3}$$

Da für die Ermittlung von W_E nur der wahrscheinlichste Pfad interessiert (Erkennung mittels Viterbi-Algorithmus [Vit73]), entartet das Summenzeichen über alle Phonemketten Ph zum Maximumoperator.

$$W_E = \operatorname*{argmax}_W \max_{Ph} \left[P(O|Ph) \cdot P(Ph|W) \cdot P(W)\right] = \operatorname*{argmax}_W \max_{Ph} P(O, Ph, W) \tag{2.4}$$

Gl. (2.4) kann in einem ‚top-down'-Ansatz implementiert werden. Ein Wortkettengenerator produziert mögliche Wortketten W mit deren a-priori-Wahrscheinlichkeit $P(W)$. Ein nachfolgender Phonemkettengenerator erzeugt aus jeder Wortkettenhypothese zugehörige Phonemketten Ph mit jeweiliger bedingter Wahrscheinlichkeit $P(Ph|W)$. Der akustische Vergleich ermittelt zu jeder generierten Phonemkette Ph und der gegebenen Beobachtungsfolge O die bedingte Wahrscheinlichkeit $P(O|Ph)$. Die Wortkette aus derjenigen Kombination aus einer Wortkette W, einer Phonemkette Ph und der gegebenen

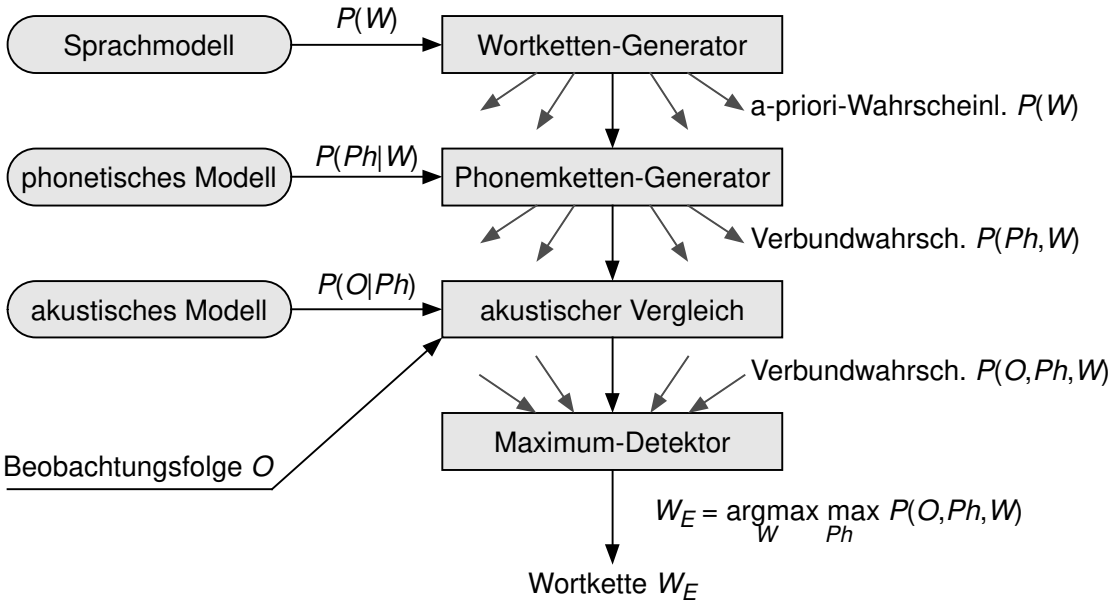


Abb. 2.6: Prinzip des ‚top-down'-Ansatzes bei der Wortkettendecodierung

Beobachtungsfolge O, welche Gl. (2.4) maximiert, gilt als erkannt und wird als die erkannte Wortkette W_E bezeichnet. Bei diesem Ansatz werden drei stochastische Wissensbasen (Modelle) zur Ermittlung der relevanten Wahrscheinlichkeiten benötigt:

- Das **Sprachmodell** muß für eine Wortkette $W = w_1 w_2 \ldots w_{nw} \ldots w_{NW}$ deren Auftrittswahrscheinlichkeit $P(W)$ abschätzen. Im Fall eines Bigramm-Sprachmodells wird diese Auftrittswahrscheinlichkeit durch das Produkt der bedingten Wahrscheinlichkeiten aus der Abfolge zweier hintereinanderfolgender Worte approximiert:

$$P(W) = P(w_1 w_2 \ldots w_{nw} \ldots w_{NW}) \approx \prod_{nw=1}^{NW} P(w_{nw} | w_{nw-1}) \tag{2.5}$$

  (In vorausgegangener Gleichung repräsentiert w_0 das Startsymbol.) Würde anstatt der Bigramm-Wahrscheinlichkeit $P(w_{nw}|w_{nw-1})$ eine k-gramm-Wahrscheinlichkeit $P(w_{nw}|w_{nw-1}, w_{nw-2}, \ldots, w_{nw-k+1})$ mit $k \geq NW - 1$ verwendet, könnte $P(W)$ exakt berechnet werden. Beim Einsatz von Bigramm-Sprachmodellen, welche die Wörter in Klassen einteilen, werden die bedingten Wahrscheinlichkeiten $P(w_{nw}|w_{nw-1})$ wiederum approximiert:

$$P(w_{nw}|w_{nw-1}) \approx P(w_{nw}|C_{nw}) \cdot P(C_{nw}|C_{nw-1}) \tag{2.6}$$

  (Hierbei repräsentiert C_0 das Startsymbol.) Die Anzahl der vorkommenden Klassen ist dabei geringer als die Anzahl der vorkommenden Worte. Ein Bigramm-Sprachmodell enthält also die Klassenübergangswahrscheinlichkeiten einer Klasse C_{nw} in Abhängigkeit von der vorausgegangenen Klasse C_{nw-1}. Unter der (stark angenäherten) Voraussetzung, daß innerhalb einer Klasse jedes Wort gleichwahrscheinlich und in der Klasse C_{nw} die Anzahl der Worte $A_{C_{nw}}$ ist, gilt:

$$P(w_{nw}|C_{nw}) = \frac{1}{A_{C_{nw}}} \tag{2.7}$$

  Somit läßt sich die Auftrittswahrscheinlichkeit für die Wortkette W folgendermaßen ausdrücken:

$$P(W) \approx \prod_{nw=1}^{NW} \frac{P(C_{nw}|C_{nw-1})}{A_{C_{nw}}} \tag{2.8}$$

- Das **phonetische Modell** enthält alle dem System bekannten Worte mit jeweiliger Aussprache als Phonemkette. Diese Wissensbasis wird daher auch als „Aussprachelexikon“, „Lexikon“ oder „Vokabular“ bezeichnet und liefert die bedingte Wahrscheinlichkeit $P(Ph|W)$ für das Auftreten einer aus NP Phonemen bestehenden Phonemkette $Ph = Ph_1 Ph_2 \ldots Ph_{nw} \ldots Ph_{NW} = ph_1 ph_2 \ldots ph_{NP}$, wenn eine vorgegebene Wortkette $W = w_1 w_2 \ldots w_{nw} \ldots w_{NW}$ vorliegt. Jedes Wort w_{nw} korrespondiert dabei zu einer Phonemkette Ph_{nw}, wobei die Bildung einer Phonemkette Ph_{nw} ausschließlich vom zugehörigen Wort w_{nw} abhängig ist. Im Gegensatz zur Wortkette sind die Pho-

nemketten Ph_{nw} lückenlos zur gesamten Phonemkette Ph aneinandergefügt, so daß die Information über vorkommende Wortgrenzen verloren geht.

Existieren zu einem Wort mehrere Phonemketten, so spricht man von Aussprachevarianten dieses Wortes. Sollten in einem vereinfachten phonetischen Modell keine Aussprachvarianten zulässig sein (d.h. zu jedem Wort existiert nur eine zulässige Phonemkette), entarten sämtliche Wahrscheinlichkeiten zu Eins und die Wahrscheinlichkeit $P(Ph|W)$ nimmt, da zu einer gegebenen Wortkette nur eine bestimmte Phonemkette korrespondiert, stets den Wert Eins an.

$$P(Ph|W) = P(Ph_1 Ph_2 \ldots Ph_{NW} \big| w_1, w_2, \ldots, w_{NW}) \approx \prod_{nw=1}^{NW} P(Ph_{nw} \big| w_{nw}) \qquad (2.9)$$

- Das **akustische Modell** liefert die bedingte Wahrscheinlichkeit $P(O|Ph)$ für das Auftreten einer Beobachtungsfolge $O = o_1, o_2, \ldots, o_{NO}$ unter der Voraussetzung einer Phonemkette $Ph = ph_1 ph_2 \ldots ph_{NP}$. Im hier beschriebenen ‚top-down'-Ansatz bedeutet dies, daß die Wahrscheinlichkeit der gegebenen Beobachtungsfolge O unter der Voraussetzung einer Phonemketten-Hypothese Ph abgeschätzt wird.

  Da einerseits die zeitliche Struktur von Sprache, andererseits die akustische Realisierung eines Phonems (durch Allophone) enorm variabel sein kann, ist das Schließen von einer Phonemkette auf eine Beobachtungsfolge ein sehr diffiziler Vorgang. Dazu haben sich in der Spracherkennung statistische Verfahren bewährt. Die drei meistverfolgten Ansätze dazu sind:

  – Dynamische Programmierung [Ney84][Rus94]
  – Neuronale Netze [Cic93][Gra92][Kin92][Mül90][Rei96]
  – Hidden-Markov-Modelle (HMMe) [Hua90][Rab89]

  Auf die ersten beiden Verfahren soll an dieser Stelle nicht näher eingegangen werden. Die in der vorliegenden Arbeit verwendeten HMMe ermöglichen durch eine Überlagerung zweier stochastischer Prozesse (Übergänge und Emissionen) eine gute Modellierung sowohl der variablen Zeitstruktur als auch der allophonischen Eigenheiten von Sprache. Da die Aneinaderreihung der Phoneme (und damit auch der nachzubildenden Allophone) in sequentieller Abfolge geschieht, kann das HMM zu einer Kettenstruktur, die nur in eine Richtung durchlaufen werden kann, vereinfacht werden. Jedes Phonem wird stellvertretend für akustische Eigenheiten zu Beginn, in der Mitte und am Ende in ein Anfangs-, ein Mittel- und ein Endsegment aufgeteilt. Jedes dieser Segmente wird im HMM von zwei Zuständen repräsentiert, wobei jeder Zustand Z_i einen Merkmalsvektor $\overrightarrow{o_{no}}$ emittiert. Übergänge von einem Zustand Z_i sind nur zum selben Zustand Z_i, zum nächsten Zustand Z_{i+1} oder zum übernächsten Zustand Z_{i+2} erlaubt.

  Die Wahrscheinlichkeit für einen Übergang von einem Zustand Z_i zu einem nachfolgenden Zustand Z_j wird mit der **Übergangswahrscheinlichkeit a_{ij}** bezeichnet:

$$a_{ij} = P(\text{Übergang nach } Z_j \big| Z_i) \qquad (2.10)$$

  Die Wahrscheinlichkeit für die Emission eines Merkmalsvektors $\overrightarrow{o_{no}}$ vom Zustand Z_i wird mit **Emissionswahrscheinlichkeit b_{ino}** bezeichnet:

$$b_{ino} = P(\text{Emission von } \overrightarrow{o_{no}} \Big| Z_i) \tag{2.11}$$

In Abb. 2.7 ist die vorher beschriebene Struktur eines Phonems als HMM (Phonemmodell) aufskizziert [Zün91].

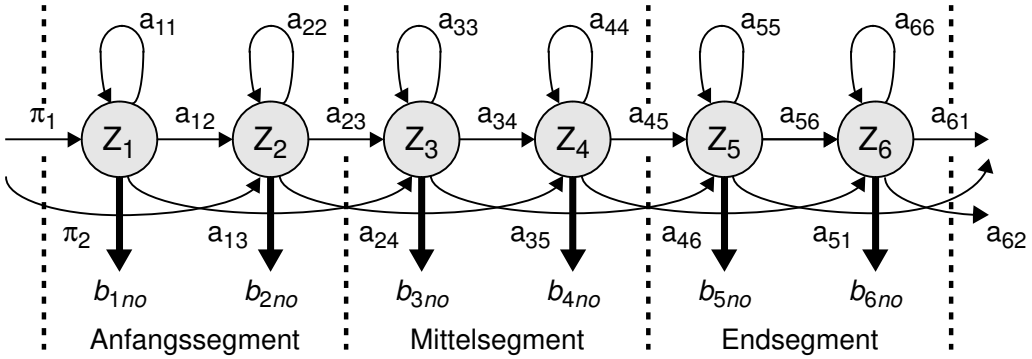


Abb. 2.7: Phonemmodell

Im Rahmen der hier beschriebenen Arbeiten wurden ausschließlich kontextunabhängige Phonemmodelle werwendet. Kontextunabhängige Phonemmodelle betrachten alle Wahrscheinlichkeiten nur in Abhängigkeit vom aktuellen Phonem. Im Gegensatz dazu berücksichtigen kontextabhängige Phonemmodelle außerdem das Vorgänger- und das Nachfolgerphonem, d.h. jedes Phonem wird im Kontext seiner beiden Nachbar-Phoneme modelliert. Allerdings muß die Parameter-Anzahl bei kontextabhängigen Modellen im Vergleich zu kontextunabhängigen Modellen mit drei potenziert werden, woraus eine wesentliche Vergrößerung des notwendigen akustisch-phonetischen Trainingsmaterials resultiert, um eine ausreichend genaue Abschätzung aller Parameter zu gewährleisten.

2.2 Einstufiger Ansatz

Ansätze, welche die semantische Decodierung innerhalb einer Stufe vollbringen, werden in der aktuellen Forschung erstaunlicherweise selten verfolgt. Dies hängt sicher damit zusammen, daß innerhalb eines Ansatzes Probleme der Signalverarbeitung, Spektralanalyse, Phonetik, Linguistik, Wissensrepräsentation und der Wissensverarbeitung bewältigt werden müssen. Ein in sich geschlossener, einstufiger Ansatz vermeidet allerdings Inkonsistenzen zwischen mehreren Modulen. Es können mehrere Wissenbasen, welche die Zusammenhänge zwischen den jeweiligen Repräsentationsebenen enthalten, im Sinne einer gesamten, integrierten Wissensbasis mit allen zur Verfügung stehenden Parametern zusammenwirken. Eventuell auftretende Fehler des ersten Moduls (im allgemeinen Wortketten-Decodierung) können sich nicht mehr auf ein nachfolgendes, davon losgelöstes Modul (semantische Analyse) auswirken.

2.2.1 Beispiele aus der aktuellen Forschung

- **Umwandlung spanischer Zahlen im Millionen-Bereich**
  Prieto und Vidal wandeln gesprochene spanische Zahlen in die korrespondierende Zif-

fernschreibweise um [Pri92]. Die Zahlen werden hierbei nicht als Einzelworte gespeichert, sondern setzen sich entsprechend einer speziellen Grammatik zusammen. Diese Umwandlung geschieht innerhalb eines einzigen Verarbeitungsschrittes mit einem sogenannten Wandler (*Transducer*), einer formalen Überführungsprozedur einer formalen Sprache (Beobachtungsfolge) in eine andere (Ziffernfolge). Diese Aufgabe ist deswegen komplex, da spanische Zahlen auf viele, teils unregelmäßige Weisen ausgesprochen werden können.

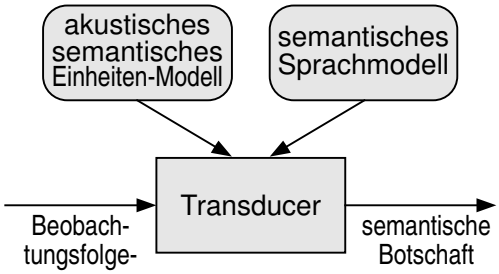


Abb. 2.8: Blockdiagramm einer einstufigen Umwandlung (übersetzt nach Prieto und Vidal [Pri92])

Der sogenannte ECGI-Algorithmus [Rul89] konstruiert eine reguläre Grammatik oder einen endlichen Zustandsautomaten, welcher diese Grammatik akzeptiert. Damit werden semantische Einheiten (*semantic units*) direkt mit akustischen Merkmalen, der Beobachtungsfolge, in Verbindung gesetzt. Das semantische Sprachmodell setzt die semantischen Einheiten zu einer gesamten semantischen Botschaft (*semantic message*) zusammen.

- **System für geographische Auskünfte in spanischer Sprache**
  Das Problem des mehrstufigen Ansatzes wird von Prieto et al. durch eine integrierte semantische Decodierung (*integrated architecture*) effizient vermieden [Pri94]. Einen ähnlichen Ansatz in gleicher Domäne verfolgen Bonafonte et al. [Bon95][Bon96]. Ein sog. integriertes Modell mit einer Gesamt-Wissensbasis dient zur direkten Umwandlung einer Beobachtungsfolge in eine semantische Botschaft. Das integrierte Modell setzt sich wie in folgender Abb. 2.9 gezeigt aus den drei Wissensbasen Phonemmodell, semantisches Einheitenmodell und semantisches Sprachmodell zusammen.

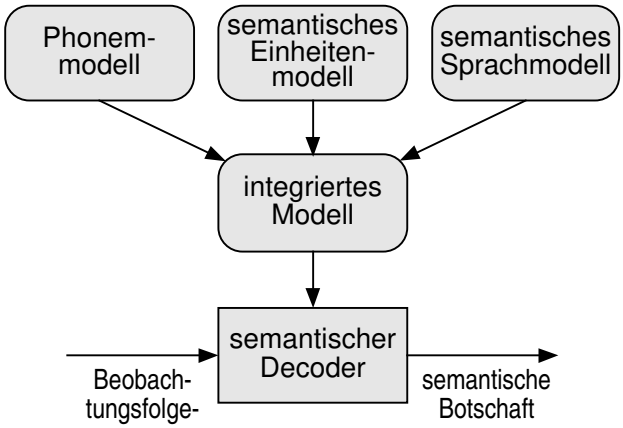


Abb. 2.9: Blockdiagramm der integrierten semantischen Decodierung (übersetzt nach Prieto et al. [Pri94])

- Das **Phonemmodell** (*phoneme model*) enthält die Beziehungen zwischen den Merkmalsvektoren und phonetischen Transkriptionen.
- Das **semantische Einheitenmodell** (*semantic unit model*) faßt die Beziehungen zwischen den phonetischen Transkriptionen und den sogenannten semantischen Einheiten zusammen.
- Das **semantische Sprachmodell** (*semantic language model*) beinhaltet die Beziehungen innerhalb der semantischen Repräsentation, die sich aus den semantischen Einheiten zusammensetzt.

2.2.2 Semantische Decodierung mit stochastischem Ansatz

Analog zur rein statistischen Wortketten-Decodierung (siehe Kap. 2.1.2) kann auch die semantische Decodierung rein statistisch in einem ‚top-down'-Ansatz ablaufen. Dazu ist die syntaktisch-semantische Repräsentationsebene in den geschlossenen Ansatz miteinzubeziehen. Alle Gleichungen sowie die ‚top-down'-Struktur sind diesbezüglich zu erweitern. Es muß jedoch ein probabilistisch quantifizierbarer Zusammenhang zwischen der syntaktisch-semantischen Repräsentation und der korrespondierenden Wortkette einer Äußerung bestehen. Aus diesem Grund wurde die semantische Gliederung S als syntaktisch-semantische Repräsentationsebene eingeführt (siehe Kapitel 3), welche es ermöglicht, eine Wahrscheinlichkeit $P(W|S)$ für das Auftreten einer zur semantischen Gliederung S gehörenden Wortkette W zu bestimmen.

Der semantische Decoder ermittelt aus einer vorliegenden Beobachtungsfolge O diejenige erkannte semantische Gliederung S_E, welche unter Vorliegen genau dieser Beobachtungsfolge O und den zur Verfügung stehenden Wissensbasen die höchste Wahrscheinlichkeit $P(S|O)$ besitzt:

$$S_E = \underset{S}{\operatorname{argmax}}\, P(S|O) \tag{2.12}$$

Diese Gleichung kann mit der Bayes'schen Regel umgeformt werden. Die sich ergebende a-priori-Wahrscheinlichkeit $P(O)$ braucht für die Maximierung nicht berücksichtigt werden, da sie bei gegebener Beobachtungsfolge O konstant ist.

$$S_E = \underset{S}{\operatorname{argmax}}\, \frac{P(O|S) \cdot P(S)}{P(O)} = \underset{S}{\operatorname{argmax}}\, \left[P(O|S) \cdot P(S)\right] \tag{2.13}$$

Die direkte Bestimmung von $P(O|S)$ ist aufgrund der Vielfalt möglicher Kombinationen aus O und S nicht ohne weiteres möglich. Deshalb werden die Wortebene W und die Phonemebene Ph als weitere Repräsentationsebenen eingeführt.

$$S_E = \underset{S}{\operatorname{argmax}} \sum_{\text{alle } W} \sum_{\text{alle } Ph} \left[P(O|Ph) \cdot P(Ph|W) \cdot P(W|S) \cdot P(S)\right] \tag{2.14}$$

Da für die Ermittlung von S_E nur der wahrscheinlichste Pfad interessiert (Erkennung mittels Viterbi-Algorithmus [Vit73]), entarten die Summenzeichen über allen Wortketten W und allen Phonemketten Ph zu Maximumoperatoren.

$$S_E = \underset{S}{\operatorname{argmax}} \max_{W} \max_{Ph} \left[P(O|Ph) \cdot P(Ph|W) \cdot P(W|S) \cdot P(S) \right] = \\ = \underset{S}{\operatorname{argmax}} \max_{W} \max_{Ph} P(O, Ph, W, S) \tag{2.15}$$

Gl. (2.15) kann zweckmäßig in einem einstufigen ‚top-down'-Ansatz implementiert werden. Ein Gliederungsgenerator produziert mögliche semantische Gliederungen mit deren a-priori-Wahrscheinlichkeit $P(S)$. Daraus erzeugt ein Wortketten-Generator mögliche Wortketten mit der zugehörigen Wahrscheinlichkeit $P(W|S)$. Ein nachfolgender Phonemketten-Generator erzeugt aus jeder Wortkettenhypothese die zugehörigen Phonemketten mit jeweiliger bedingter Wahrscheinlichkeit $P(Ph|W)$. Durch akustischen Vergleich wird zu jeder vorliegenden Phonemkette die bedingte Wahrscheinlichkeit $P(O|Ph)$ für die vorliegende Beobachtungsfolge O ermittelt. Die semantische Gliederung derjenigen Kombination aus einer semantischen Gliederung S, einer Wortkette W, einer Phonemkette Ph und der gegebenen Beobachtungsfolge O, welche Gl. (2.15) maximiert, gilt als erkannt und wird als die erkannte semantische Gliederung S_E bezeichnet.

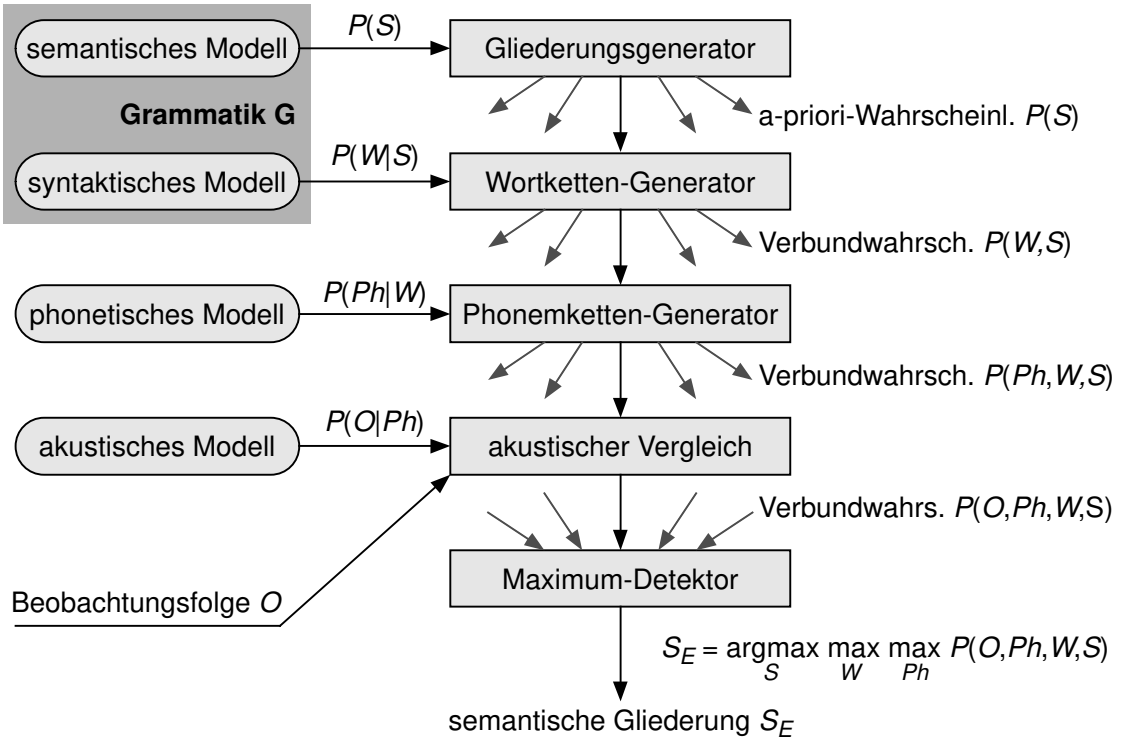


Abb. 2.10: Prinzip des ‚top-down'-Ansatzes bei der semantischen Decodierung

Bei diesem Ansatz werden vier stochastische Wissensbasen (Modelle) zur Ermittlung der relevanten Wahrscheinlichkeiten benötigt:

- Das **semantische Modell** liefert die a-priori-Wahrscheinlichkeit $P(S)$ für das Auftreten einer semantischen Gliederung S. Eine genaue Beschreibung dieser Wissensbasis wird in Kapitel 4.1 gegeben.

- Das **syntaktische Modell** liefert die bedingte Wahrscheinlichkeit $P(W|S)$ für das Auftreten einer Wortkette W gegeben eine semantische Gliederung S. Eine genaue Beschreibung dieser Wissensbasis wird in Kapitel 4.2 gegeben.
- Das **phonetische Modell** liefert die bedingte Wahrscheinlichkeit $P(Ph|W)$ für das Auftreten einer Phonemkette Ph gegeben eine Wortkette W. Diese Wissenbasis ist prinzipiell mit derjenigen bei der Wortketten-Decodierung identisch – eine genaue Beschreibung steht im Kapitel 2.1.2.
- Das **akustische Modell** liefert die bedingte Wahrscheinlichkeit $P(O|Ph)$ für das Auftreten einer Beobachtungsfolge O gegeben eine Phonemkette Ph. Diese Wissenbasis kann unverändert von der Wortketten-Decodierung übernommen werden – eine genaue Beschreibung steht im Kapitel 2.1.2.

Mit allen vier Wissensbasen ist es unter der Annahme der statistischen Unabhängigkeit möglich, die zur Maximierung benötigte Verbundwahrscheinlichkeit $P(O, Ph, W, S)$ durch Multiplikation der jeweiligen bedingten Wahrscheinlichkeiten zu berechnen.

Durch die beiden Wissensbasen des semantischen und des syntaktischen Modells wird eine Grammatik für die vom System interpretierbaren Wortketten festgelegt [Bau95] [Sta95]. Diese Grammatik kann im weitesten Sinne als kontextfrei betrachtet werden [Sta97b].

2.3 Mehrstufiger versus einstufiger Ansatz

Der meistverbreitete Ansatz zur semantischen Decodierung einer Beobachtungsfolge ist ein zweistufiger Ansatz mit der Wortebene als Schnittstelle. Zwischen den hierbei eingesetzten Modulen Wortkettendecodierung und semantischer Analyse wird grundsätzlich ‚bottom-up' vorgegangen. Wird jedoch bei der Wortkettendecodierung ein Fehler begangen, kann dieser von der nachfolgenden semantischen Textanalyse nur in wenigen Fällen korrigiert werden. Außerdem steht bei der Wortkettendecodierung nur akustisches, phonetisches und sprachliches – nicht jedoch semantisches – Wissen zur Verfügung. Gerade mit dem semantischen Wissen lassen sich aber sinnvolle Einschränkungen bezüglich in Frage kommender Wortketten treffen. Eine Wortkette kann zwar durchaus im Sinne der Bigramm-Grammatik des Wortkettendecoders korrekt, jedoch semantisch vollkommen sinnlos und damit für die weitere Verarbeitung unbrauchbar sein.

Im Extremfall könnte man sich die semantische Decodierung als dreistufigen ‚bottom-up'-Prozeß, der aus Phonemdecoder, Phonem-Graphem-Umsetzung und semantischer Textanalyse besteht, vorstellen. Sollten die zur Verfügung stehenden Wissensbasen stochastischer Natur sein, könnten bei der nun beschriebenen Vorgehensweise lediglich die jeweiligen Wahrscheinlichkeiten einzeln für sich maximiert werden:

$$\begin{aligned} Ph_E = \underset{Ph}{\operatorname{argmax}}\, P(Ph|O) \;\rightarrow\; W_E = \underset{W}{\operatorname{argmax}}\, P(W|Ph_E) \;\rightarrow \\ S_E = \underset{S}{\operatorname{argmax}}\, P(S|W_E) \end{aligned} \tag{2.16}$$

Obwohl dies auf den ersten Blick naheliegend erscheinen mag, ist obige Gl. (2.16) für den praktischen Einsatz völlig unzureichend, da hierbei nur begrenztes Wissen unabhängig voneinander verwendet wird. Es ist sicherlich auch der menschlichen Verarbeitung entsprechender, daß konkurrierendes Wissen mehrerer Ebenen (semantisches, syntaktisches, morphologisches, phonetisches, akustisches) gleichzeitig für den Decodierungsprozeß zur Verfügung steht.

Aus diesen Gründen liegt ein einstufiger Ansatz auf der Hand, bei dem alle zur Verfügung stehenden Wissensbasen mit semantischem, syntaktischem, phonetischem und akustischem Wissen gleichzeitig und konsistent zusammenwirken können. Somit kann eine aus den aufmultiplizierten Wahrscheinlichkeiten aller Wissensbasen bestehende Verbundwahrscheinlichkeit über alle Repräsentationsebenen gebildet werden, die gemeinsam, ‚top-down' und innerhalb eines einzigen Rechenschrittes maximiert werden kann:

$$S_E = \operatorname*{argmax}_S \max_W \max_{Ph} \left[P(O|Ph) \cdot P(Ph|W) \cdot P(W|S) \cdot P(S) \right] \tag{2.17}$$

Der dazu nötige Suchraum ist natürlich um ein Vielfaches größer, als beim mehrstufigen ‚bottom-up'-Ansatz. Es müssen ja eine Vielzahl von Hypothesen aus semantischer Gliederung, Wortkette und Phonemkette bezüglich einem vorliegenden Merkmalsvektor verwaltet werden. Da natürlich aus Gründen des begrenzten Speicherplatzes und der limitierten Rechenkapazität nicht alle denkbaren Hypothesen verwaltet werden können, müssen Hypothesen mit einer geringen Wahrscheinlichkeit zu gegebener Zeit verworfen werden (sogenanntes *Pruning*). Der auf dem ‚top-down'-Ansatz basierende Suchalgorithmus nach der erkannten semantischen Gliederung S_E ist ausführlich in [Sta96] beschrieben.

Kapitel 3

Darstellung sprachlicher Bedeutung mit der semantischen Gliederung

Das zentrale und wichtigste Kriterium von gesprochener und geschriebener Sprache liegt in ihrer Bedeutsamkeit, nämlich der Tatsache, daß Sprache und Text Bedeutung haben und daß es möglich ist, durch Sprechen oder Schreiben etwas mitzuteilen. Ohne diesen Zweck wären Sprache und Text völlig nutzlos, da in der Regel niemand rein bedeutungslose Laute sprechen oder bedeutungslose Buchstabenkombinationen schreiben würde.

Die menschliche Informationsaufnahme von Sprache und Text bedient sich in der Regel des auditiven bzw. des visuellen Kanals. In Ausnahmefällen wird auch der taktile Kanal benutzt, zum Beispiel für Blindenschrift. Die primär aufgenommene optische, akustische oder haptische Information (im Sinne des quantitativen Informationsaspekts[9]) ist jedoch nur zur Übertragung oder Speicherung relevant, befriedigt jedoch nicht das Interesse des Informationsempfängers. Für diesen ist nur der Bedeutungsinhalt bzw. die Intention die letztendlich wissenswerte Information (im Sinne des qualitativen Informationsaspekts).

3.1 Was ist ‚Semantik‘?

Unter dem Begriff *Semantik* versteht man allgemein die Bedeutung von Zeichen[10] und Zeichenfolgen. Im folgenden seien damit ausschließlich sprachliche Zeichen (Wort bzw. Wortkette als Repräsentation von gesprochenen oder sprechbaren Phonemen) gemeint. Dabei ist jedem Zeichen eine Ausdrucks- und eine Inhaltsseite zugeordnet. Die Ausdrucksseite beinhaltet dabei die korrekte Wortbildung (Morphologie) sowie die korrekte und verständliche Aussprache des Wortes (Phonologie). Das sprachliche Zeichen verbindet jedoch nicht eine Sache und einen Namen miteinander, sondern vielmehr eine *konzeptuelle* und eine *akustische* Vorstellung. Die konzeptuelle Vorstellung ist dabei nicht ein Gegenstand selbst, sondern eine Abstraktion von derartigen denkbaren Gegenständen.

9) Die Informationstheorie nach Shannon-Wiener [Mar66][Sha49] macht eine quantitative Informationsaussage.

10) Auch einem Bild kann eine Bedeutung (im Sinne von Semantik) zugeordnet werden.

Die akustische Vorstellung ist keine laut ausgesprochene Phonemkette, sondern eine psychologische Vorstellung einer derartigen Lautkette. Beim Menschen sind akustische und konzeptuelle Vorstellung durch Assoziation unlösbar miteinander verknüpft, so daß er in der Lage ist, zwischen diesen Vorstellungen in Sekundenbruchteilen hin- und herzuschalten.[11)] Selbst wenn die vom Gehör aufgenommene Sprache von Störgeräuschen, grammatikalischen Fehlern oder spontansprachlichen Effekten überlagert ist, kann der Mensch relativ mühelos eine konzeptuelle Vorstellung entwickeln. Genau diese Umschaltung bereitet einer Computerapplikation größte Schwierigkeiten, und zwar aus den folgenden Gründen:

- Die Bedeutung eines oder mehrerer Worte kann durchaus mehrdeutig sein, wobei die erforderliche Disambiguisierung durch Einbeziehung des Kontexts erfolgt. Diese Mehrdeutigkeit kann mit der „Kontextsensitivität von Sprache“ erklärt werden, da sehr oft die pragmatische Umgebung eines Wortes zu dessen korrekter semantischer Decodierung benötigt wird. Die dazu erforderliche enorm große Wissensbasis muß sich ein Mensch innerhalb eines lebenslangen Lernprozesses durch Erziehung, Erfahrung, Beobachtung, Interesse und Unterricht aneignen.
- Die robuste Erkennung gesprochener Sprache ist enorm rechen- und speicherplatzaufwendig und wird derzeit noch nicht hinreichend beherrscht. Dabei auftretende Erkennungsfehler können eine fehlerhafte semantische Decodierung zur Folge haben. Störgeräusche, spontansprachliche Effekte oder unvollständiges Vokabular (dazu siehe Kapitel 9) verursachen in der Regel weitere Erkennungsfehler.
- Die Akquisition und Repräsentation von sprachlichem Wissen stellt ein wissenschaftlich noch nicht vollständig erforschtes Gebiet dar. In diesem Zusammenhang ist bislang ungeklärt, auf welche Weise das menschliche Gehirn aufgenommene Information speichert.
- Die Sprachproduktion gelingt nur bei eng umgrenztem Sachverhalt. Meist werden ausgegebene natürlichsprachliche Äußerungen nicht wirklich ‚online‘ generiert, sondern mittels einfacher Inferenz von einer Datenbank abgerufen.

Aus diesen Gründen erscheint beim derzeitigen Stand der Technik ein System, das alle denkbaren, gesprochenen Eingaben verstehen und richtig darauf reagieren soll, nicht realisierbar. Sollte also ein System gesprochene Sprache automatisch verstehen können, müssen die zu verarbeitenden, natürlichsprachlichen Benutzereingaben aus einer klar umgrenzten Domäne kommen. Dies stellt zwar eine gewaltige Vereinfachung dar, wird jedoch problemlos akzeptiert, da bei einem Grafikprogramm oder einer Fahrplanauskunft eben nur Äußerungen innerhalb der jeweiligen Domäne zu erwarten sind.

---

11) Als Beispiel sei das Wort „Stuhl“ näher betrachtet: Sobald ein Mensch die Phonemkette /ʃtu:l/ akustisch wahrnimmt, kann er zur konzeptuellen Vorstellung umschalten. Die Vorstellung, wie ein Gegenstand aussehen und beschaffen sein muß, um ein Stuhl zu sein, fällt jedem Menschen leicht. Auch wenn ein Mensch einen Stuhl sieht oder fühlt, ist das Umschalten zur akustischen Vorstellung einfach: Sofort ist er in der Lage, sich die Phonemkette /ʃtu:l/ vorzustellen, diese korrekt auszusprechen und sogar noch Synonyme wie „Sitzgelegenheit“ oder „Hocker“ zu bilden. Es werden noch weitere Assoziationen gebildet: Wie schwer ein Stuhl ist, wozu man ihn brauchen kann, ob er schön, ästhetisch oder bequem ist, wieviel er kostet usw.

3.2 Repräsentation von Semantik

Die Darstellung von sprachlicher Bedeutung ist nicht nur als formal-theoretische Disziplin interessant, sondern auch als Zwischenebene innerhalb einer sprachverstehenden oder sprachübersetzenden Applikation notwendig. Dazu haben sich linguistische Verfahren einerseits, mathematisch-logische Verfahren andererseits etabliert. Zur Semantik-Repräsentation definiert Winston mehrere aufschlußreiche Unterscheidungskriterien [Win87]:

- **Institutionsbasierte Semantik:** Hierbei liegt eine informative Beschreibung des Sachverhaltes vor, die nicht unter formal-logischen Gesichtpunkten erstellt wurde und keine fest definierte Semantik enthält.
- **Äquivalenzsemantik:** Hierbei werden die Beschreibungen des darzustellenden Sachverhaltes mit Beschreibungen einer anderen Darstellung in Beziehung gebracht, die eine fest definierte Semantik aufweist, zum Beispiel die Umschreibung mittels Prädikatenlogik. (Auch die Umschreibung mit einer formal-logischen, syntaktisch-semantischen Repräsentation innerhalb einer sprachverstehenden Applikation wäre im Sinne einer solchen Äquivalenzsemantik.)
- **Prozedurale Semantik:** Wenn eine Menge von Programmen gegeben ist, die aufgrund der Beschreibungen der Repräsentationssprache arbeiten, dann wird die Semantik durch den Ablauf oder das Ergebnis dieser Programme definiert. (Die Reaktion eines sprachverstehenden Grafikeditors wäre genau im Sinne einer prozeduralen Semantik: Denn das Resultat, was am Bildschirm angezeigt wird, ist ja der der Äußerung zugrundeliegende Bedeutungsinhalt.)
- **Deskriptive Semantik:** Hierbei wird die darzustellende Semantik mittels Beschreibungen in natürlicher Sprache wiedergegeben.

Im Sinne der eben angesprochenen Äquivalenzsemantik muß eine formal-logische Repräsentationsebene gefunden werden, um die Bedeutung sprachlicher Eingaben wiederzugeben. Dabei sollten semantisch gleichbedeutende Wortketten (z.B. „schiebe die kugel nach links“ bzw. „den ball bitte nach links rücken“) möglichst eine identische semantische Repräsentation besitzen. Vereinfachend sei diese Repräsentationsebene vollkommen unabhängig vom aktuellen Wirkungsfeld, d.h. vom pragmatischen Einfluß, und unabhängig von vorausgehenden oder künftigen Äußerungen. Ob nun im Kontext der vorigen Wortketten die Kugel wirklich existiert, die nach links geschoben werden soll, spielt für diese semantische Repräsentation keine Rolle – entscheidend ist einzig die vorliegende Äußerung.

Klassische Notationen des Bedeutungsinhaltes bzw. der Struktur von Sprache und Text sind die Prädikatenlogik, die Darstellung mit semantischen Netzen und die Valenz- bzw. Dependenzgrammatik, welche in den folgenden Kapiteln kurz dargestellt werden, um damit eine Brücke zu der im Rahmen dieser Arbeit entwickelten semantischen Gliederung zu schlagen.

3.2.1 Prädikatenlogik

Seit langem wird die Prädikatenlogik in der Sprachphilosophie zur Repräsentation natürlichsprachlicher Information benutzt [Kow79][Lov78][Pin93][Sag90]. An dieser Stelle sei nur die Prädikatenlogik erster Ordnung mit folgendem Inventar betrachtet:

- Logische Konstanten
  – Junktoren: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (hat zur Folge)
  – Quantoren: $\forall$ (für alle gilt), $\exists$ (für ein existierendes gilt)
  – Identitätszeichen
- Individuenvariablen a, b, c, d, ...
- Nicht-logische Konstanten
  – Individuenkonstanten
  – n-stellige Relationskonstanten für $n \geq 0$

Bei der einfachsten Form der semantischen Analyse natürlichsprachlicher Ausdrücke werden zunächst die Wörter aus einer Wortkette kategorisiert, indem ihnen nicht-logische Konstanten eines bestimmten Typs zugeordnet werden. Dem Objekt rote kugel wird die Individuenvariable a zugeordnet, die durch die verknüpften einstelligen Relationskonstante (kugel(a) $\wedge$ rot(a)) spezifiziert wird. Die Wortart (z.B. Substantiv, Verb, Adjektiv) ist dabei völlig unabhängig vom logischen Typ des semantischen Wertes. Das zur Verfügung stehende Typenspektrum der Prädikatenlogik wird in der natürlichen Sprache nicht vollständig ausgenutzt, da zur Umschreibung natürlicher Sprache nur selten vier- oder mehrstellige Relationskonstanten zu finden sind. Im folgenden sind beispielhaft einige Relationskonstanten zur Grafikeditor-Domäne aufgelistet:

- 1-stellige Relationskonstanten:
  neu(x), löschen(x), kugel(x), quader(x), rot(x), gelb(x), groß(x), klein(x)
- 2-stellige Relationskonstanten:
  färben(x, y), skalieren(x, y), neben(x, y)
- 3-stellige Relationskonstanten:
  zwischen(x, y, z)

Beispiele für die Darstellung von einigen Wortketten aus der Domäne „Grafikeditor“ mittels Prädikatenlogik wären:

erzeuge eine große rote kugel
neu(a) $\rightarrow$ kugel(a) $\wedge$ groß(a) $\wedge$ rot(a) $\wedge$ 1(a)

erzeuge zwischen dem quader und dem zylinder eine gelbe kugel
neu(a) $\rightarrow$ kugel(a) $\wedge$ gelb(a) $\wedge$ 1(a) $\wedge$ zwischen(a, b, c) $\wedge$ $\exists b$(quader(b)) $\wedge$ $\exists c$(zylinder(c))

erzeuge zwei kugeln und drei würfel
(neu(a) $\wedge$ neu(b)) $\rightarrow$ kugel(a) $\wedge$ 2(a) $\wedge$ würfel(b) $\wedge$ 3(b)

vergrößere alle grünen zylinder um das dreifache
skalieren(a, b) $\rightarrow$ $\forall a$(zylinder(a) $\wedge$ grün(a)) $\wedge$ faktor3(b)

schiebe den zylinder neben der roten kugel neben den braunen kegel
$\text{schieben}(a, b) \rightarrow \exists a(\text{zylinder}(a) \wedge \exists c(\text{kugel}(c) \wedge \text{rot}(c)) \wedge \text{neben}(a, c))$
$\wedge \exists d(\text{kegel}(d) \wedge \text{braun}(d)) \wedge \text{neben}(b, d)$

erzeuge zwei quader und färbe alle weißen kugeln gelb
$[\text{neu}(a) \rightarrow \text{quader}(a) \wedge 2(a)] \wedge [\text{färben}(b, c) \rightarrow \forall b(\text{kugel}(b) \wedge \text{weiß}(b)) \wedge \text{gelb}(c)]$

lösche beide zylinder auf der kugel oder den kegel
$(\text{löschen}(a) \vee \text{löschen}(b)) \rightarrow \forall a(\text{zylinder}(a) \wedge \exists c(\text{kugel}(c)) \wedge \text{über}(a, c)) \wedge \exists b(\text{kegel}(b))$

lösche alle objekte mit ausnahme des lilanen zylinders
$\text{löschen}(a) \rightarrow \forall a(\neg\, (\text{zylinder}(a) \wedge \text{violett}(a)))$

Angesichts dieser Beispiele scheint die Umwandlung einer sprachlichen Anweisung in die korrespondierende prädikatenlogische Notation relativ einfach. Mit Hilfe dieser Notation lassen sich kontrollierte Werkzeuge zur weiteren formal-logischen Auswertung (Induktion, Deduktion, Komposition, Substitution) [Pin93] einsetzen, falls dies erforderlich sein sollte. Da ein unmittelbarer Zusammenhang zur jeweiligen Wortkette nicht gegeben scheint, ist die Angabe einer Wahrscheinlichkeit, mit welcher einer Wortkette von der korrespondierenden prädikatenlogischen Darstellung abhängt, nicht möglich.

3.2.2 Semantische Netze

Semantische Netze werden in der Künstlichen Intelligenz als Repräsentation sprachlichen Wissens eingesetzt. Ein semantisches Netz benennt Objekte und bezeichnet Beziehungen zwischen Objekten [Win87]. Die Syntax eines solchen SNes ist im Grunde genommen trivial. Es existieren:

- Objekte, die graphisch durch benannte Rechtecke dargestellt werden. Sie bilden die Knoten des semantischen Netzes.
- Beziehungen zwischen Objektpaaren, die graphisch durch benannte Pfeile dargestellt werden. Sie bilden die Kanten des semantischen Netzes. Diese Kanten beschreiben einen Knoten durch sinnvolle Bezeichnungen (z.B.: Form, Farbe, ist ein, Teil von).

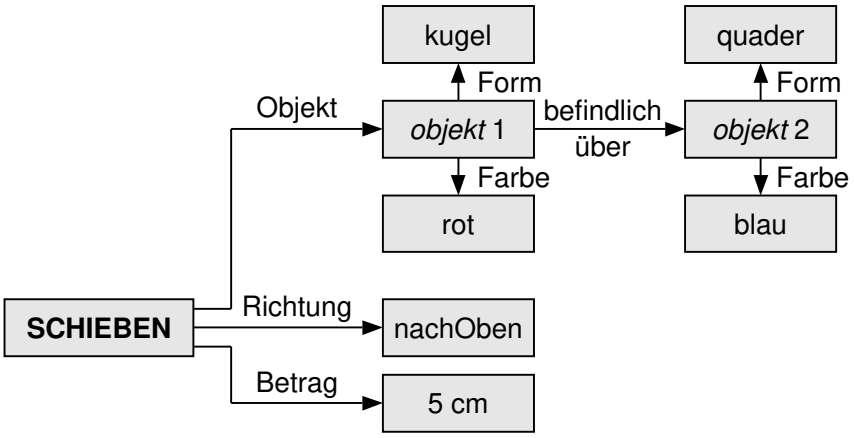


Abb. 3.1: Semantisches Netz der Äußerung „schiebe die rote kugel über dem blauen quader fünf zentimeter nach oben“

- Allgemeingültige Behauptungen (z.B. „alle“, „jede“), die das semantische Netz als Formel auffassen und mittels des Allquantor-Symbols ‚$\forall$ ‘ für alle denkbaren Fälle referenzieren.

In Abb. 3.1 ist ein solches semantisches Netz, welches zu einer denkbaren Äußerung aus der Grafikeditor-Domäne korrespondiert, aufgezeichnet. Darin wird eine zunehmende Spezifizierung, ausgehend von dem zentralen Schiebe-Objekt, in der von den Kanten vorgegebenen Richtung deutlich. Dies ermöglicht auch eine Variabilität bezüglich Art und Anzahl denkbarer Spezifikationen eines vorgegebenen Objektes.

3.2.3 Valenz- bzw. Dependenzgrammatik

Die Valenz- oder Dependenzgrammatik geht im Gegensatz zur Konstituenten- oder Phrasenstrukturgrammatik [Hau93][Lin94] über das Kriterium der syntaktischen Form hinaus. Sie bezeichnet ein Abhängigkeitsverhältnis sprachlicher Elemente innerhalb eines Satzes und geht davon aus, daß Sätze (in unserem Fall Wortketten) eine hierarchische und damit nichtlineare Struktur haben, die durch eine Abhängigkeitsrelation festgelegt ist [Gör88]. Diese hierarchische Satzstruktur baut sich nicht allein aus der syntaktische Form des Satzes auf, sondern auch aus semantischen Charakteristika der einzelnen Worte. Die zentrale Annahme ist dabei, daß das Verb den gesamten Prozeß ausdrückt und das Zentrum des Satzes ist. Von ihm können weitere Nachfolger abhängen (*Dependenz* = Abhängigkeit). Die dazu benötigte Fähigkeit der Verben, eine bestimmte Anzahl und nur bestimmte Arten von Nachfolgern binden zu können, kann mit der Wertigkeit eines chemischen Elements verglichen werden und wird daher *Valenz* genannt. Analog dazu sind die Verben klassifizierbar in solche mit einer, zwei, drei usw. Valenzen [Tes65].

Auf der zweiten Hierarchieebene sind dem Verb die *Aktanten* untergeordnet, das sind diejenigen Dinge, die in irgendeiner Weise am Prozeß, der durch das Verb ausgedrückt wird, beteiligt sind. Morphologisch sind die Aktanten stets Substantive oder deren Platzhalter (Pronomina). Auf gleicher Stufe, also auch direkt vom Verb abhängig, finden sich die *Zirkumstanten* oder Umstandsangaben. Sie erläutern die Art und Weise, wie der Prozeß abläuft und sind daher Adverben oder adverbiale Ausdrücke. Im allgemeinen werden auf der zweiten Ebene die Aktanten links und die Zirkumstanten rechts angeordnet. Die dritte Hierarchieebene beinhaltet sogenannte *Indizes*, wie zum Beispiel Artikel, Adjektive oder adjektivische Pronomina der Aktanten. In folgender Abb. 3.2 seien einige Beispiele von Äußerungen aus der Grafikeditor-Domäne gezeigt, deren Satzstruktur aufgrund der herrschenden Dependenz aufgezeigt sind:

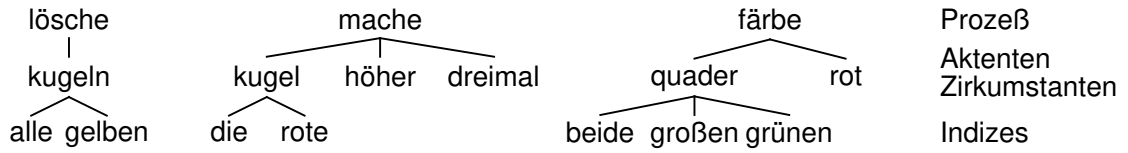


Abb. 3.2: Struktur-Darstellungen von Äußerungen mittels Dependenzgrammatik

Es müssen nicht alle Valenzen mit Aktanten oder Zirkumstanten gefüllt werden, es können also auch Leerstellen freigelassen werden. Das Verb kann nicht die Reihenfolge der

Konstituenten festlegen. Die Valenz ist eine abstrakte Eigenschaft, daher kann die Abhängigkeitsbeziehung nur über das Vorkommen an sich, nicht aber über die Position des Vorkommens entscheiden. Weiterhin ist die Valenz des Verbs eine semantische Gegebenheit, von der jedoch ganz entscheidend die dem Satz zugrundeliegende Syntax abhängt, da die Aktanten und Zirkumstanten auch syntaktisch dem Verb zuzuordnen sind [Pel75].

3.3 Definition der semantischen Gliederung

Zum Einsatz innerhalb einer sprachverstehenden Applikation wird zum Lösen der Gl. (2.15) eine formale Repräsentation des Bedeutungsinhaltes einer natürlichsprachlichen gesprochenen oder geschriebenen Äußerung gesucht, die

1. **wortnah** genug ist, um einen direkten und probabilistischen Bezug zur jeweiligen Wortkette zu ermöglichen[12)],
2. **hierarchisch aufgebaut** ist, um eine Struktur nach festen (auch rekursiven) Regeln zu ermöglichen und um bestehende Abhängigkeiten probabilistisch zu erfassen[13)],
3. **formal-logisch korrekt** ist, im Sinne, daß der Bedeutungsinhalt konsistent und logisch nachvollziehbar repräsentiert wird,
4. **generalisierend** ist, daß semantisch äquivalente, aber unterschiedliche Wortketten identisch repräsentiert werden und
5. **maschinennah** genug ist, um diese formale Repräsentation mit möglichst einfachen Mechanismen in maschineninterpretierbare Kommandos umzuwandeln.

Dazu wird die semantische Gliederung S als semantische Repräsentation einer gesprochenen Äußerung, die aus einer eng umgrenzten Domäne (im zunächst betrachteten Fall der ‚Grafikeditor') stammt und keinen Nebensatz aufweist, eingeführt [Mül94][Sta94]. Sie kann aufgefaßt werden als eine baumartige und damit hierarchische Struktur, die sich aus N kleineren bedeutungstragenden Einheiten zusammensetzt, welche im folgenden *semantische Untereinheiten* (oder kurz *Semune*) s_n genannt werden:

$$S = \{s_1, s_2, \ldots, s_n, \ldots, s_N\} \tag{3.1}$$

Jedes Semun s_n kann mit (X+2) Komponenten durch seinen Typ $t[s_n]$, seinen Wert $v[s_n]$ und Verweise auf seine $X \geq 1$ Nachfolger

$$q_1[s_n], \ldots, q_x[s_n], \ldots, q_X[s_n] \in \{s_{n+1}, \ldots, s_N, \text{ leer}\} \tag{3.2}$$

beschrieben werden:

12) Für die in Gl. (2.15) dargestellte ‚top-down'-Decodierung ist die Wahrscheinlichkeit $P(W|S)$ notwendig, d.h. die probabilistische Beziehung einer Wortkette W zu dieser noch zu definierenden semantischen Repräsentation S.

13) Für die in Gl. (2.15) dargestellte ‚top-down'-Decodierung ist die Wahrscheinlichkeit $P(S)$ notwendig, d.h. die a-priori-Wahrscheinlichkeit dieser semantischen Repräsentation S.

$$\boxed{s_n = \left(t[s_n], v[s_n], q_1[s_n], \ldots, q_X[s_n] \right)} \text{ mit } X \geq 1 \tag{3.3}$$

- Der **Typ** $t[s_n]$ gibt die Anzahl X der Nachfolger fest vor[14)] und schränkt die Menge möglicher Typen $t[q_1[s_n]], \ldots, t[q_X[s_n]]$ dieser Nachfolger-Semune ein. Außerdem trifft er eine sinnvolle Auswahl möglicher ihm zuzuordnender Werte $v[s_n]$.
- Der **Wert** $v[s_n]$ gibt in der Regel die eigentliche Bedeutung des Semuns s_n an.
- Jeder **Nachfolger** $q_x[s_n]$ spezifiziert einen bestimmten Sachverhalt des Semuns s_n.
  - Ist eine solche Spezifizierung in der Äußerung vorhanden, ist dieser Nachfolger $q_x[s_n]$ mit einem weiteren Semun innerhalb der semantischen Gliederung identisch. Der Nachfolger wird in diesem Fall als *Nachfolger-Semun* $q_x[s_n]$ bezeichnet.

    $$q_x[s_n] \in \{s_{n+1}, \ldots, s_N\} \tag{3.4}$$

  - Sollte in der Äußerung die diesbezügliche Spezifizierung nicht vorkommen, wird auf einen *leeren Nachfolger* $q_x[s_n]$ verwiesen. In diesem Fall gilt:

    $$q_x[s_n] = \text{leer} \tag{3.5}$$

    Um die in Kap. 4.1.1 beschriebene Folgewahrscheinlichkeit konsistent zu beschreiben, wird einem leeren Nachfolger der Typ ‚leer', jedoch kein Wert zugeordnet.

    $$t[\text{leer}] = \text{leer} \tag{3.6}$$

  Die Typen sämtlicher Nachfolger $q_1[s_n], \ldots, q_x[s_n], \ldots, q_X[s_n]$ des Semuns s_n können vereinfacht als **Nachfolgerschar** $t_q(s_n)$ bezeichnet werden. Die Anordnung der einzelnen Nachfolger ist dabei informationstragend und darf nicht wahlfrei verändert werden. Innerhalb einer Nachfolgerschar $t_q(s_n)$ ist eine Mischung von Nachfolger-Semunen und leeren Nachfolgern erlaubt.

$$t_q(s_n) = \left(t[q_1[s_n]], \ldots, t[q_x[s_n]], \ldots, t[q_X[s_n]] \right) \text{ mit } X \geq 1 \tag{3.7}$$

Ein Semun s_n mit den Verweisen auf X Nachfolger $q_1[s_n], \ldots, q_X[s_n]$ wird graphisch wie in folgender Abbildung dargestellt, wobei die Nachfolger stets von oben nach unten mit 1 bis X indiziert werden. Der Verweis auf ein Nachfolger-Semun wird durch die Kante „⟶" markiert. Im Gegensatz dazu kennzeichnet die Kante „⟙" den Verweis auf einen leeren Nachfolger.

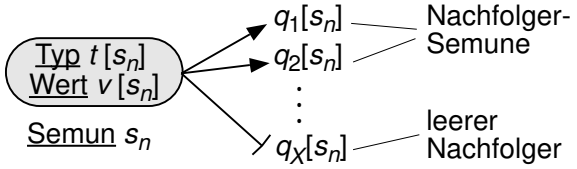


Abb. 3.3: Darstellung eines Semuns s_n mit X Nachfolgern

14) Das heißt: $X = X(t[s_n])$. Derzeit werden Semune mit $1 \leq X \leq 5$ betrachtet.

Die gesamte semantische Gliederung S bildet dabei einen ‚Baum' mit dem Semun s_1 als ‚Wurzel' und den leeren Nachfolgern als ‚Blätter'. Alle Semune $s_2, \ldots, s_N$ besitzen genau ein Vorgänger-Semun.

Ein Semun $s_n \in S$ mit (rekursiv betrachtet) allen seinen Nachfolger-Semunen bis hin zu allen abschließenden leeren Nachfolgern bildet einen **Ast**, der mit $S(s_n)$ bezeichnet wird. Die Semune innerhalb jedes Astes $S(s_n)$ sind fortlaufend von n beginnend indiziert.

$$S(s_n) = \{s_n, s_{n+1}, s_{n+2}, \ldots\} \tag{3.8}$$

Jeder Ast bildet somit eine Teilmenge der semantischen Gliederung:

$$S(s_n) \subset S \quad \text{für} \quad n \neq 1 \tag{3.9}$$

Entsprechend zu den vorangegangenen Betrachtungen ist der Ast $S(s_1)$ des Wurzelsemuns s_1 identisch zur gesamten semantischen Gliederung S.

$$S(s_1) = S \tag{3.10}$$

Folgende Abbildung zeigt als Beispiel den Baum einer aus N=11 Semunen bestehenden semantischen Gliederung S.

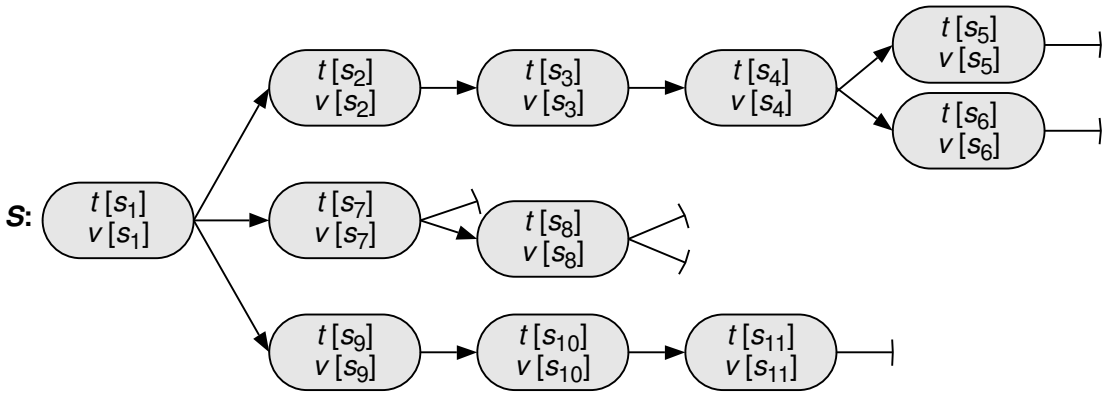


Abb. 3.4: Verknüpfung mehrerer Semune s_n zum Baum einer semantischen Gliederung S

In obigem Fall ist X=3 für s_1, X=2 für s_4, s_7 und s_8 sowie X=1 für alle anderen Semune (Verweise auf leere Nachfolger zählen ebenfalls). Der Ast $S(s_2) = \{s_2, s_3, s_4, s_5, s_6\}$ würde aus fünf Semunen, der Ast $S(s_4) = \{s_4, s_5, s_6\}$ aus drei und der Ast $S(s_7) = \{s_7, s_8\}$ aus zwei Semunen bestehen.

Jedes Semun repräsentiert einen Teil der Bedeutung der gesamten Äußerung. Die Bedeutung eines Semuns wird durch die ihm zugeordneten Nachfolger, die in einer festen Beziehung zueinander stehen, genauer spezifiziert. Jeder Ast $S(s_n) \subset S$ repräsentiert einen für sich geschlossenen Teil des Bedeutungsinhaltes.

Ein **Linie** ist eine direkte Abfolge von m+1 Semunen s_n, …, s_{n+m} innerhalb eines Astes mit jeweils genau X=1 Nachfolger. Eine Linie beginnt mit dem Nachfolger-Semun eines Semuns mit $X \geq 2$ Nachfolgern und wird von einem Semun, welches einen (d.h. X=1) leeren Nachfolger besitzt, oder vom Vorgänger-Semun eines Semuns mit $X \geq 2$ Nachfolgern beendet. Innerhalb einer semantischen Gliederung wird eine Linie aus den Semunen s_n, …, s_{n+m} mit dem Index-Tupel $\binom{n}{n+m}$ bezeichnet. Die Menge aller vorkommender Linien in der semantischen Gliederung S wird in der Menge $L(S)$ zusammengefaßt. In obiger Abbildung 3.4 existieren demnach zwei solcher Linien

$$L(S) = \left\{ \binom{2}{3}, \binom{9}{11} \right\} . \tag{3.11}$$

Die Anzahl der vorhandenen Linien ergibt sich trivial zu

$$\left| L(S) \right| = 2 . \tag{3.12}$$

In diesem Zusammenhang sei festgestellt, daß die Anordnung der Semune innerhalb einer Linie auf den von der semantischen Gliederung ausgewiesenen Bedeutungsinhalt keinen Einfluß hat – im Klartext:

Die Anordnung der Semune innerhalb einer Linie ist wahlfrei!

In obiger semantischer Gliederung S wäre zum Beispiel die Anordnung der Semune s_2 und s_3 beliebig, d.h. entweder →(s_2)→(s_3)→ oder →(s_3)→(s_2)→. Entsprechendes gilt ebenfalls für die Semun-Anordnung innerhalb der anderen Linie →(s_9)→(s_{10})→(s_{11})⊣.

Es ist zu beachten, daß innerhalb der Abfolge der Nachfolger die Anordnung der Semune informationstragend ist und nicht verändert werden darf. So würde ein Vertauschen des Astes $S(s_2)$ mit dem Ast $S(s_7)$ die Bedeutung des Semuns s_1 und damit diejenige der gesamten semantischen Gliederung S verändern.

Zwei semantische Gliederungen gelten als *äquivalent*, wenn sie die gleiche Information beinhalten. Eine äquivalente semantische Gliederung entsteht (abgesehen von Beschränkungen des syntaktischen Modells) durch Permutationen einzelner Semune innerhalb einer Linie.

Im Prinzip ähnelt der Aufbau stark an die Strukturdarstellung nach der Valenz- oder Dependenzgrammatik (siehe Kap. 3.2.3). Das jeweils „regierende“ Befehlsverb bildet als zentraler Teil die Wurzel der semantischen Gliederung. Ein einzelnes Semun kann im weitesten Sinne als X-stellige prädikatenlogische Relationskonstante angesehen werden [Pin93], wobei eine nullstellige Relationskonstante durch X=1 Nachfolger des Typs ‚leer‘ modelliert wird. Die Verknüpfung einzelner Semune zur semantischen Gliederung unterscheidet sich jedoch wesentlich von der Darstellung des Bedeutungsinhaltes mittels Prädikatenlogik. Wenn auch mathematisch nicht so exakt, bietet die semantische Gliederung folgende Vorteile:

- Die semantische Gliederung S ist eine wortnahe Darstellung des Bedeutungsinhaltes und ermöglicht eine unmittelbare und probabilistische Beziehung zu einer Wortkette W, wobei mit entsprechenden Wissensbasen die bedingte Wahrscheinlichkeit $P(W|S)$ ermittelt werden kann – siehe dazu das folgende Kapitel 3.4. Sie berücksichtigt lediglich den semantischen Inhalt der Äußerung ohne Berücksichtigung der pragmatischen Konstellation. Da die semantische Gliederung kein Wissen über vorher gesprochene Äußerungen besitzt, können Ellipsen[15], anaphorische[16] oder kataphorische[17] Verweise nicht aufgelöst werden – dies muß in einer übergeordneten Instanz geschehen.
- Zur Verknüpfung der Semune gibt es nur einen Mechanismus, nämlich die Kennzeichnung weiterer Semune als sogenannte Nachfolger. Beim Entwurf der Modelle zur Berechnung der a-priori-Wahrscheinlichkeit $P(S)$ muß daher nur diese eine Art der Verknüpfung betrachtet werden.

3.4 Bezug zur korrespondierenden Wortkette

Die semantische Gliederung ist eine wortnahe Darstellung des Bedeutungsinhaltes einer Äußerung. Ihr Aufbau hängt unter anderem von der Wortwahl und, wenn auch bedingt, von der Wortstellung der zugrundeliegenden Wortkette ab. Trotzdem besitzt sie die Fähigkeit zur Generalisierung: Unterschiedliche, jedoch bedeutungsgleiche Wortketten korrespondieren zu identischen semantischen Gliederungen. Dazu werden folgende Festlegungen getroffen:

- Jedes Wort w_{nw} aus der Wortkette W korrespondiert zu genau einem Semun s_n aus der semantischen Gliederung S.
- Jedem Semun s_n der semantischen Gliederung S wird genau ein bedeutungstragendes Wort $w^+[s_n]$ und maximal ein bedeutungsloses Wort $w^-[s_n]$ aus der Wortkette W zugeordnet. Dabei können unter Umständen zwei oder mehr Worte (im morphologischen Sinn) zu einem zusammenhängenden „Wort“ kombiniert werden. Da lediglich die korrespondierende semantische Gliederung als Endresultat von Bedeutung ist, kann auf eine morphologisch korrekte Wortrepräsentation verzichtet werden. Der Entwurf des syntaktischen Modelles (Bestimmung von $P(W|S)$, der Wahrscheinlichkeit für das Auftreten einer Wortkette W für eine vorliegende semantische Gliederung S) ist somit ohne weitere Repräsentationsebenen möglich.
- Jeder Ast $S(s_n) \subseteq S$ korrespondiert zu einer zusammenhängenden Teilwortkette $W(s_n) \subseteq W$.

Als Beispiel sei in Abb. 3.5 der Bezug zwischen der Wortkette „bitte den schönen grünen quader neben der ähm roten kugel löschen“ und deren semantischen Gliederung dargestellt. Auf das bedeutungstragende Wort eines Semuns sei mit dem Pfeil „⟶“ hinge-

15) Eine Ellipse ist eine Form der Textverknüpfung, wobei der Textverweis durch Leerstellen erzeugt wird, z.B. die Äußerung „jetzt etwas dunkler“ im Anschluß an „mache die kugel größer und heller“.

16) Rückverweis, z.B. die Äußerung „mache ihn grün“ im Anschluß an „erzeuge einen kegel“.

17) Vorverweis, wurde jedoch beim Systemeinsatz niemals beobachtet.

wiesen, auf das optionale bedeutungslose Wort („bitte“, „schönen“, „ähm“) mit dem Pfeil „⟶“. Die zu dem jeweiligen Ast $S(s_n)$ gehörende, lückenlose Teilwortkette $W(s_n)$ wird darunter aufgezeigt. Dabei ist zu beachten, daß sich die Anordnung dieser Teilwortketten aus der Hierarchie der zugrundeliegenden Semune ableitet.

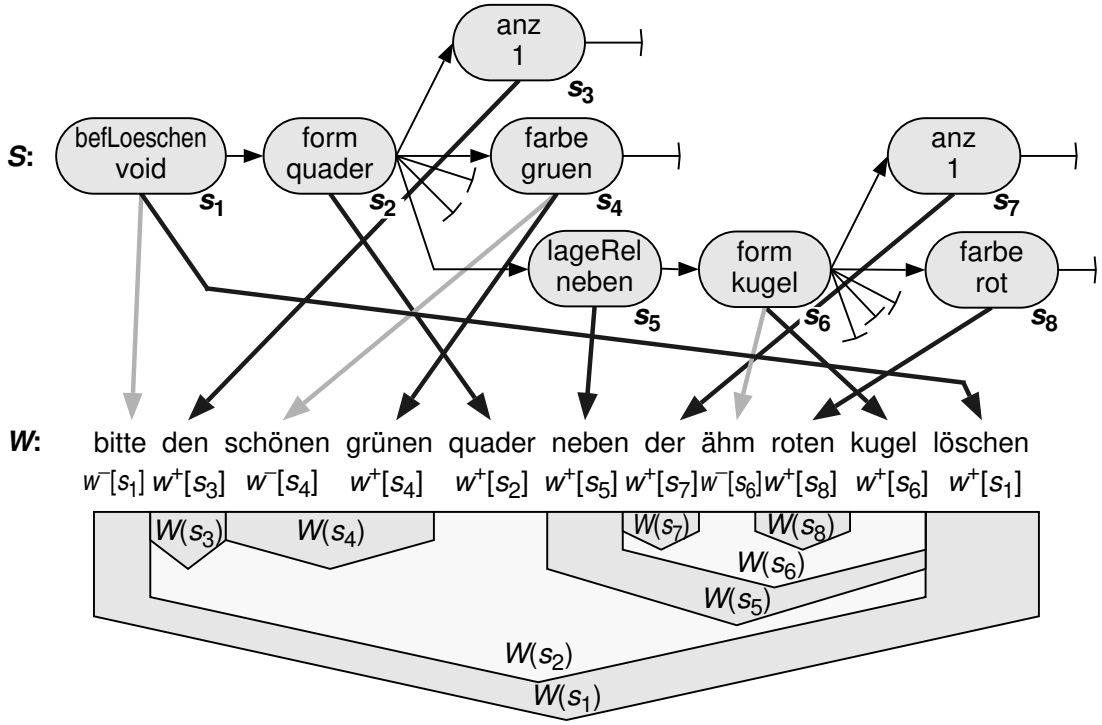


Abb. 3.5: Semantische Gliederung S und dazu erzeugte Teilwortketten $W(s_n)$ in W

3.5 Eingrenzung der darzustellenden Semantik

Da die Domäne auf einen umgrenzten Anwendungsbereich beschränkt ist, sollten auch diejenigen Bedeutungsinhalte, die den vom Benutzer gegebenen Anweisungen zugrundeliegen, aus einer endlichen Menge stammen und intuitiv aus der vorgegebenen Aufgabe ableitbar sein. Ziel ist nun, einen zu erwartenden Bedeutungsinhalt so in kleine Portionen (z.B. bestimmte Befehle, Formen, Farben, usw.) zu unterteilen, daß sich mit einem möglichst kleinen semantischen Inventar alle innerhalb der betrachteten Domäne denkbaren und sinnvollen Bedeutungsinhalte darstellen lassen.

Wie man bei der Entwicklung einer neuen Programmiersprache zu überlegen hat, welche Kommandos im jeweiligen (Sprach-)Inventar enthalten sein müssen, ist nun zu überlegen, welche Intentionen ein Benutzer bei der Interaktion mit einer domänenspezifischen, sprachverstehenden Applikation verfolgen und wie er seine Intention semantisch mit natürlichsprachlichen Äußerungen ausdrücken könnte. Das in diesem Fall zu entwickelnde Inventar ist beim Einsatz der semantischen Gliederung die Menge aus den zur Verfügung stehenden Typen und Werten. Durch sinnvolle Kombinationen dieser Typen und Werte untereinander sowie aus verschiedenen Typen, Werten und Nachfolgern gebildeter Semune lassen sich theoretisch unendlich viele semantische Gliederungen mit einem relativ geringen Typen- und Werteinventar erzeugen.

3.6 Auswahl der Typen und Werte

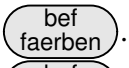
Ausschließlich bei der Festlegung der Typen und Werte sowie deren Beziehung zur Wortkette steckt im beschriebenen Ansatz zur Semantikdecodierung pragmatisches (domänenspezifisches) und linguistisches (sprachspezifisches) Wissen. Sämtliche darauf aufbauende Algorithmen sind rein stochastischer Natur, sämtliche Entscheidungen bei der semantischen Decodierung werden nur aufgrund von Wahrscheinlichkeiten getroffen.

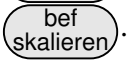
3.6.1 Pragmatischer Gesichtpunkt

Die sinnvolle Auswahl der Typen und Werte geschieht zunächst heuristisch nach pragmatischen Gesichtspunkten. Die als semantische Gliederungen darzustellenden Äußerungen sollen ausschließlich aus einer umgrenzten Domäne stammen. Es muß daher ein geeignetes Typen- und Werteinventar gebildet werden, welches nur den Bedeutungsinhalt von zu erwartenden Äußerungen abdeckt. Für Äußerungen außerhalb der betrachteten Domäne sind keine Typen und Werte zu definieren. (Es wird weder von einem sprachverstehenden Grafikprogramm eine Zugauskunft verlangt werden, noch soll ein sprachverstehender Videorecorder eine Telefonverbindung herstellen.)

3.6.2 Linguistischer Gesichtspunkt

Um den Suchraum bei der semantischen Decodierung [Sta97b] klein zu halten und um mögliche Fehlinterpretationen zu vermeiden, sollte das semantische Modell möglichst wenige Hypothesen bilden. Dies kann dadurch effektiv eingeschränkt werden, indem die Vielfalt möglicher Nachfolger eines bestimmten Typs gering gehalten wird. Dies soll an folgendem Beispiel erläutert werden: Ursprünglich wurden für alle vorkommenden Befehle zum Ändern eines Objektes (z.B. färben, skalieren, verschieben, drehen) der Typ „bef“ gewählt:

- Semun für den Befehl zum Färben eines Objekts: 

.
- Semun für den Befehl zum Skalieren eines Objekts: 

.

Sofern diese Befehle in eindeutige Worte wie z.B. „färbe“ bzw. „skaliere“ gekleidet wurden, war dies eine zweckmäßige und robuste Lösung. Bei der Interpretation der beiden Wortketten

W_1: mache den kegel blau
W_2: mache den kegel größer

konnten jedoch Probleme beobachtet werden. Anzahl, Form, Farbe und Größe konnten zwar richtig interpretiert werden, doch dem zum Befehl „mache“ korrespondierenden Semun mit dem Typ „bef“ wurde fälschlicherweise stets derjenige Wert zugeordnet, welcher gemäß dem semantischen Modell (siehe Kap. 4.1) die höchste Wertwahrscheinlichkeit besaß. Die Worte „blau“ und „größer“ haben auf diese Wahrscheinlichkeit – wie später gezeigt wird – keinen Einfluß. Daher haben sich die anfänglich gewählten Semune mit dem Typ „bef“ nicht bewährt.

In den Wortketten W_1 und W_2 gibt nur der Kontext einen Hinweis, welche Bedeutung dem mehrdeutigen Wort „mache“ zugrundeliegt. Es ist also essentiell wichtig, diesen durch die Worte „blau“ und „größer“ bestehenden Kontext auf das Wort „mache“ auszudehnen und das zu diesem Befehlswort korrespondierende Semun dementsprechend anzupassen. Die Folgewahrscheinlichkeit für das Auftreten einer bestimmten Nachfolgerkombination ist definitionsgemäß nur vom Typ abhängig. So ist es zweckmäßig, diese Abhängigkeit bereits bei der Typ-Deklaration zu berücksichtigen.

Mit folgenden Semunen konnten die vorher aufgetretenen Decodierungsfehler vermieden werden. Diese dem bedeutungstragenden Wort „mache“ zugrundeliegenden Semune minimieren effizient die Vielfalt ihrer möglichen Nachfolger:

- befFaerben void, dessen zweiter Nachfolger ein Semun des Typs „farbe“ ist,
- befSkalieren void, dessen zweiter Nachfolger ein Semun des Typs „groesse“ ist.

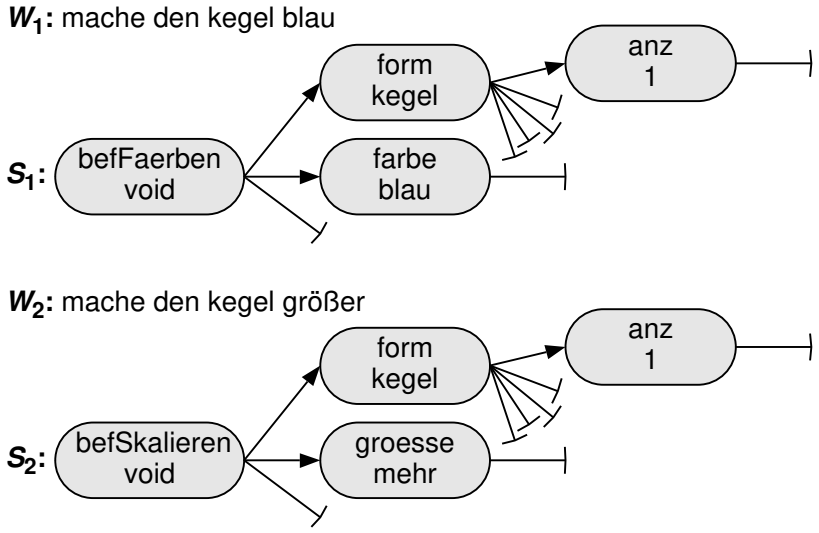


Abb. 3.6: Semantische Gliederungen S_1 und S_2, welche W_1 und W_2 repräsentieren

Derzeit werden zum Beschreiben des gesammelten Sprachmaterials aus der Domäne „Grafikeditor“ 41 Typen mit insgesamt 263 Werten eingesetzt. In Anhang wird das derzeit benutzte Typen- und Werteinventar aufgelistet.

Hat ein Semun mehrere Nachfolger, ist deren Anordnung abhängig vom Typ dieses Semuns streng festgelegt. Die Position eines Nachfolgers ist bedeutungstragend, da er die Semantik des Semuns in vorgegebener Weise spezifiziert. Eine Vertauschung hätte eine falsche semantische Interpretation zur Folge. Zwei Beispiele für die vom Semuntyp vorgegebene Nachfolger-Systematik werden in umseitiger Abb. 3.7 aufgezeigt.

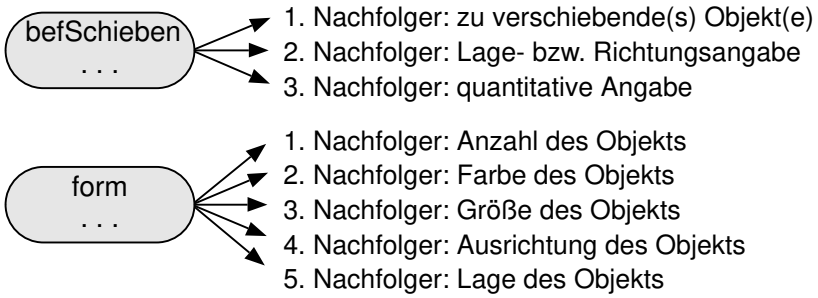


Abb. 3.7: Beispiele zweier Semune für festgelegte Nachfolgerabfolge

3.7 Varianten einer semantischen Gliederung

Eine semantische Gliederung muß in jedem Fall den der Äußerung des Benutzers zugrundeliegenden Bedeutungsinhalt korrekt und eindeutig wiedergeben. Das bedeutet, daß zu einer gegebenen semantischen Gliederung auch nur eine einzige Benutzerintention zuzuordnen sein darf und damit keine Alternativen zulässig sind. Umgekehrt betrachtet, kann prinzipiell dieselbe Intention von verschiedenen, nicht-äquivalenten semantischen Gliederungen repräsentiert werden. Dies könnte theoretisch bei den äußerlich sehr unterschiedlichen Wortketten

W_1: schiebe die linke kugel an die stelle des quaders und
W_2: schiebe das blaue objekt fünf millimeter nach oben

mit den entsprechend unterschiedlichen semantischen Gliederungen zutreffen, nämlich genau dann, falls bei einer bestimmten Konstellation die am weitesten links gelegene Kugel das einzige blaue Objekt darstellt, und diese genau fünf Millimeter unter dem Quader positioniert ist. Darüber hinaus existieren in seltenen Fällen unterschiedliche, nicht-äquivalente semantische Gliederungen, die aber denselben Bedeutungsinhalt unabhängig von der vorherrschenden Konstellation repräsentieren. Diese Gliederungen seien als *Varianten* einer semantischen Gliederung bezeichnet. Wie bereits vorher angesprochen, ist die semantische Gliederung eine wortnahe Darstellung. Um die an sie gestellten Randbedingungen betreffs des geforderten Bezugs zur korrespondierenden Wortkette einzuhalten, muß jedoch ein völlig identischer Bedeutungsinhalt in Ausnahmefällen durch unterschiedliche semantische Gliederungen ausgedrückt werden. Es seien die folgenden Wortketten betrachtet:

W_3: mache die kugel doppelt so groß
W_4: vergrößere die kugel auf das zweifache
W_5: verdopple die größe der kugel

Trotz gleichen Bedeutungsinhalts der zugrundeliegenden Äußerungen weichen die korrespondierenden semantischen Gliederungen voneinander ab; anders wären die in Kapitel 3.4 aufgelisteten Regeln über den Bezug zur korrespondierenden Wortkette nicht einzuhalten. Diese semantischen Gliederungen sind Varianten und beinhalten denselben Bedeutungsinhalt, nämlich die Kugel auf die zweifache Größe zu skalieren.

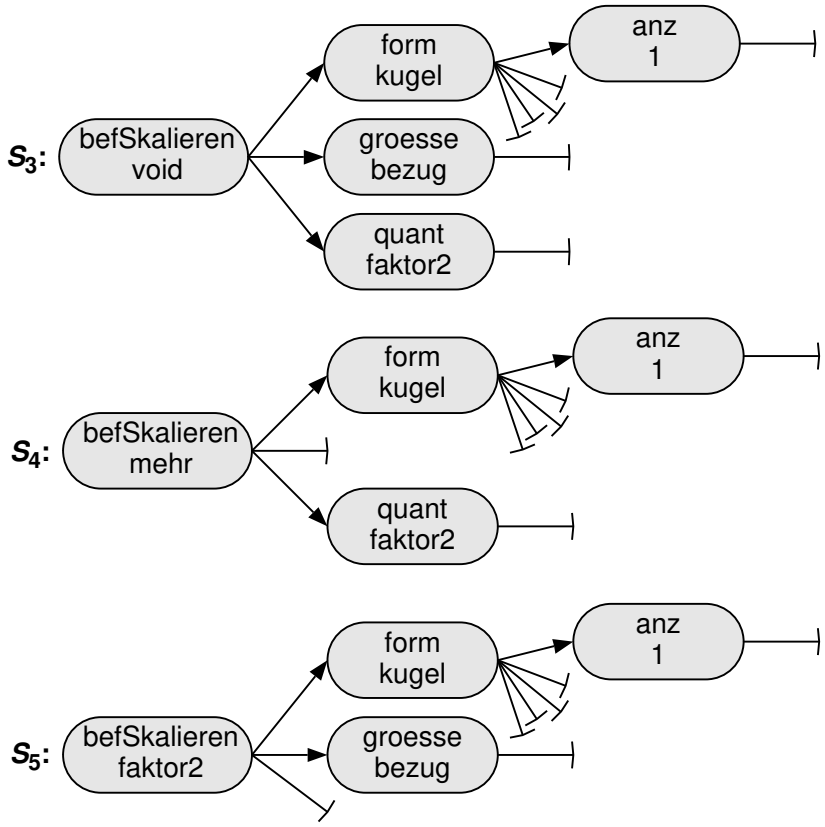


Abb. 3.8: Semantische Gliederungen S_3, S_4 und S_5 korrespondierend zu W_3, W_4 und W_5

Natürlich muß der der semantischen Decodierung nachfolgende Intentionsdecoder diese Uneinheitlichkeit erkennen und daraus eine identische Reaktion der Maschine ableiten, wodurch die Transformation in identische Benutzerintentionen letztendlich gewährleistet wird. Trotz dieser Ambiguität hat sich die semantische Gliederung als eine mächtige und universell einsetzbare Repräsentationsebene zum Einsatz innerhalb eines sprachverstehenden und sprachübersetzenden Systems für kurze, domänenspezifische Äußerungen bewährt.

Kapitel 4

Die stochastische Grammatik

Im beschriebenen Ansatz zur semantischen Decodierung kommen unter anderem die beiden folgenden stochastischen Wissensbasen zum Einsatz:

- Das **semantische Modell** liefert die a-priori-Wahrscheinlichkeit $P(S)$ für das Auftreten einer semantischen Gliederung S.
- Das **syntaktische Modell** liefert die bedingte Wahrscheinlichkeit $P(W|S)$ für das Auftreten einer Wortkette W gegeben eine semantische Gliederung S.

Mit beiden Wissensbasen zusammen läßt sich eine Verbundwahrscheinlichkeit $P(W, S)$ für das gemeinsame Auftreten einer Wortkette W zusammen mit einer semantischen Gliederung S angeben:

$$P(W, S) = P(W|S) \cdot P(S) \tag{4.1}$$

Da zu einer vorgegebenen Wortkette W in der Regel genau eine oder nur sehr wenige semantische Gliederungen S korrespondieren, kann mit Gl. (4.1) auch die a-priori-Wahrscheinlichkeit einer Wortkette W bestimmt werden:

$$P(W) = \sum_{\substack{\text{alle } S, \text{ die zu } W \\ \text{korrespondieren}}} P(W, S) \tag{4.2}$$

Eine probabilistische Aussage über die Wahrscheinlichkeit einer Wortkette W kann prinzipiell als stochastische Grammatik aufgefaßt werden. Die „erlaubten“ Wortketten nehmen dabei eine Wahrscheinlichkeit $P(W) \neq 0$, die „verbotenen“ Wortketten eine Wahrscheinlichkeit $P(W) = 0$ an.

Gl. (4.2) zeigt weiterhin, daß die Kombination von semantischem und syntaktischem Modell auch für die traditionelle Spracherkennung (im Sinne der reinen Umwandlung von gesprochener Sprache in geschriebenen Text) zur Ermittlung von $P(W)$ herangezogen werden kann. Mit diesen beiden stochastischen Wissensbasen könnte das traditionelle (üblicherweise n-gramm-) Sprachmodell ersetzt werden, womit insbesondere si-

chergestellt wäre, daß die Erkennung ausschließlich in einer für die betrachtete Domäne semantisch sinnvollen Wortkette resultiert.

4.1 Das semantische Modell

Das semantische Modell ist diejenige Wissenbasis, welche die a-priori-Wahrscheinlichkeit $P(S)$ für das Auftreten einer semantischen Gliederung S bereitstellt. Diese Wahrscheinlichkeit wird im betrachteten ‚top-down'-Ansatz zur Funktion des Gliederungs-Generators herangezogen. Die Vielfalt von S ist jedoch zu groß, um $P(S)$ direkt aus einem Trainingskorpus zu schätzen, denn es ist nicht anzunehmen, daß alle denkbaren und sinnvollen semantischen Gliederungen auch im Training gesehen werden. Das semantische Modell muß eine endliche Anzahl Parameter enthalten, die sich einerseits zuverlässig aus begrenztem Trainingsmaterial bestimmen lassen, die andererseits jedoch auf die jeweilige a-priori-Wahrscheinlichkeit $P(S)$ einer unbegrenzten Menge semantischer Gliederungen S schließen lassen.

4.1.1 Wahrscheinlichkeiten im semantischen Modell

Vorgeschlagen wird daher die Abschätzung folgender bedingter Wahrscheinlichkeiten erster Ordnung, aus denen sich die a-priori-Wahrscheinlichkeit $P(S)$ in guter Näherung bestimmen läßt:

- Die **Wurzelwahrscheinlichkeit** f_0 gibt die a-priori-Wahrscheinlichkeit an, daß das Wurzel-Semun s_1 vom Typ $t[s_1]$ ist:

$$f_0 = P(t[s_1]) \tag{4.3}$$

  Das semantische Modell muß für jeden Typ jeweils eine Wurzelwahrscheinlichkeit bereitstellen.

- Die **Wertwahrscheinlichkeit** e_n gibt die Wahrscheinlichkeit an, daß ein Semun vom Typ $t[s_n]$ den Wert $v[s_n]$ aufweist:

$$e_n = P\Big(v[s_n] \,\Big|\, t[s_n] \Big) \tag{4.4}$$

  Das semantische Modell muß für jede Kombination aus einem Typ und einem Wert jeweils eine Wertwahrscheinlichkeit bereitstellen.

- Die **Folgewahrscheinlichkeit** f_n gibt die Wahrscheinlichkeit an, daß ein Semun vom Typ $t[s_n]$ die Nachfolger-Schar $t_q(s_n) = (t[q_1[s_n]], \ldots, t[q_X[s_n]])$ besitzt:

$$f_n = P\Big(t\big[q_1[s_n]\big], \ldots, t\big[q_X[s_n]\big] \,\Big|\, t[s_n] \Big) = P\Big(t_q(s_n) \,\Big|\, t[s_n] \Big) \tag{4.5}$$

  Das semantische Modell muß für jede denkbare Kombination aus (X+1) Typen jeweils eine Folgewahrscheinlichkeit bereitstellen. Dabei ist zu beachten, daß jeder Nachfol-

ger $q_x[s_n]$ mit $1 \le x \le X$ sowohl ein Nachfolger-Semun als auch ein leerer Nachfolger sein kann. Für den Typ eines leeren Nachfolgers gilt dabei $t[\text{leer}] = \text{leer}$.

Diese im semantischen Modell enthaltenen Wahrscheinlichkeiten werden innerhalb des Trainings, welches im Kapitel 5 beschrieben wird, abgeschätzt und in den folgenden Ausführungen als bekannt betrachtet.

4.1.2 Wahrscheinlichkeit einer semantischen Gliederung

Die Bestimmung der a-priori-Wahrscheinlichkeit $P(S)$ sei anhand des unten dargestellten Ausschnitts einer semantischen Gliederung S erläutert.

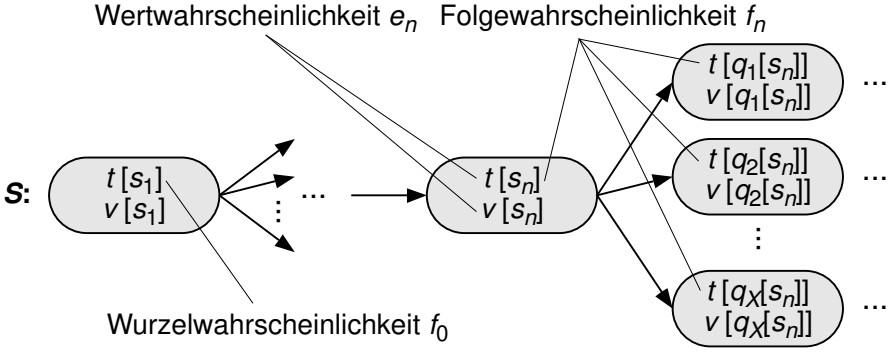


Abb. 4.1: Ausschnitt einer semantischen Gliederung S mit zugeordneten Wurzel, Wert- und Folgewahrscheinlichkeiten

- Das Wurzelsemun s_1 wird mit der nur von seinem Typ $t[s_1]$ abhängigen Wurzelwahrscheinlichkeit f_0 beaufschlagt.
- Jedes Semun s_n mit $1 \le n \le N$ wird mit der von seinem Typ $t[s_n]$ und seinem Wert $v[s_n]$ abhängigen Wertwahrscheinlichkeit e_n beaufschlagt.
- Jedes Semun s_n mit $1 \le n \le N$ wird mit der von seinem Typ $t[s_n]$ und seiner Nachfolgerschar $t_q(s_n) = (t[q_1[s_n]], \ldots, t[q_X[s_n]])$ abhängigen Folgewahrscheinlichkeit f_n beaufschlagt.

Unter der Annahme der statistischen Unabhängigkeit dieser Wahrscheinlichkeiten ergibt sich die a-priori-Wahrscheinlichkeit $P(S)$ als gemeinsames Produkt der Wurzelwahrscheinlichkeit mit den Wert- und Folgewahrscheinlichkeiten aller N Semune der semantischen Gliederung S:

$$P(S) = f_0 \cdot \prod_{n=1}^{N} (e_n \cdot f_n) \tag{4.6}$$

Diese Vorgehensweise sei nun an einem konkreten Beispiel erläutert. Gegeben sei die in folgender Abbildung gezeigte semantische Gliederung S_1, die aus N=3 Semunen besteht. Sämtliche zur Bestimmung von $P(S_1)$ relevanten Wahrscheinlichkeiten sind dabei unten stehend gezeigt.

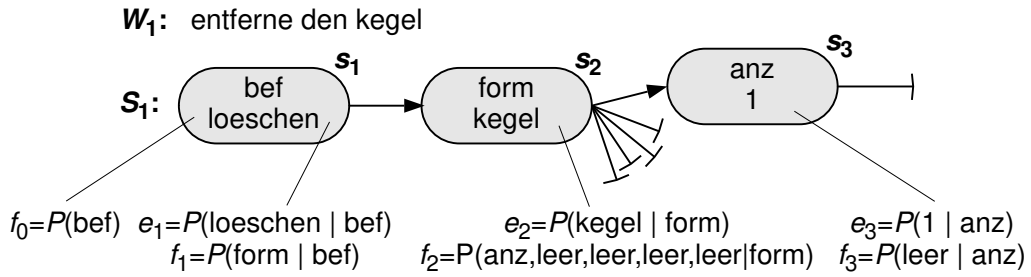


Abb. 4.2: Semantische Gliederung S_1 mit Wurzel-, Wert- und Folgewahrscheinlichkeiten

- Das Wurzelsemun s_1 wird beaufschlagt mit
  - der Wurzelwahrscheinlichkeit f_0, hier die Wahrscheinlichkeit für das Auftreten des Typs „bef“ im Wurzelsemun s_1,
  - der Wertwahrscheinlichkeit e_1, hier die Wahrscheinlichkeit für das Auftreten des Wertes „loeschen“ gegeben den Typ „bef“, und
  - der Folgewahrscheinlichkeit f_1, hier die Wahrscheinlichkeit für das Auftreten eines Nachfolger-Semuns des Typs „form“ gegeben den Typ „bef“.

$$
\begin{aligned}
f_0 &= P(t[s_1] = \text{bef}) \\
e_1 &= P(v = \text{loeschen} \,|\, t = \text{bef}) \\
f_1 &= P(t[q_1] = \text{form} \,|\, t = \text{bef})
\end{aligned}
$$

- Das Semun s_2 wird beaufschlagt mit
  - der Wertwahrscheinlichkeit e_2, hier die Wahrscheinlichkeit für das Auftreten des Wertes „kegel“ gegeben den Typ „form“, und
  - der Folgewahrscheinlichkeit f_2, hier die Wahrscheinlichkeit, daß der Typ des ersten Nachfolger-Semuns „anz“ ist und alle anderen Nachfolger leere Nachfolger (d.h. jeweiliger Typ „leer“) sind, falls der Typ „form“ vorliegt.

$$
\begin{aligned}
e_2 &= P(v = \text{kegel} \,|\, t = \text{form}) \\
f_2 &= P(t[q_1] = \text{anz}\,,\, t[q_2] = \text{leer}\,,\, t[q_3] = \text{leer}\,,\, t[q_4] = \text{leer}\,,\, t[q_5] = \text{leer} \,\big|\, t = \text{form})
\end{aligned}
$$

- Das Semun s_3 wird beaufschlagt mit
  - der Wertwahrscheinlichkeit e_3, hier die Wahrscheinlichkeit für das Auftreten des Wertes „1“ gegeben den Typ „anz“, und
  - der Folgewahrscheinlichkeit f_3, hier die Wahrscheinlichkeit für das Auftreten eines leeren Nachfolgers (d.h. Typ „leer“) gegeben den Typ „anz“.

$$
\begin{aligned}
e_3 &= P(v = 1 \,|\, t = \text{anz}) \\
f_3 &= P(t[q_1] = \text{leer} \,|\, t = \text{anz})
\end{aligned}
$$

Die Auftrittswahrscheinlichkeit $P(S_1)$ für die vorliegende semantische Gliederung S_1 ergibt sich gemäß Gl. (4.6) als Produkt dieser Wahrscheinlichkeiten, die das semantische Modell liefert:

$$P(S_1) = f_0 \cdot e_1 \cdot f_1 \cdot e_2 \cdot f_2 \cdot e_3 \cdot f_3$$

Obwohl die Annahme der statistischen Unabhängigkeit der jeweiligen bedingten Wahrscheinlichkeiten in der Praxis nicht gegeben ist, da sich kontextuelle Bindungen in natürlichen Sprachen oft über mehrere Worte erstrecken, rechtfertigt die einfache Modellierbarkeit die Anwendung von Gl. (4.6). Beliebig genau könnte $P(S)$ bestimmt werden, wenn nicht bedingte Wahrscheinlichkeiten 1. Ordnung, sondern mindestens N-ter Ordnung verwendet würden. Die Anzahl der zu trainierenden Parameter nimmt allerdings mit steigender Ordnung von N exponentiell zu, was einerseits die Trainierbarkeit erschweren würde, da wesentlich mehr Trainingsmaterial gesammelt und verarbeitet werden müßte, und andererseits bei einer späteren ‚top-down'-Decodierung der zu bearbeitende Hypothesen-Suchraum viel zu stark anwachsen würde.

Mit einem geeigneten Typeninventar lassen sich jedoch auch Abhängigkeiten höherer Ordnung erzielen, indem nur die Abfolge bestimmter Typen erlaubt ist. Ein Beispiel dazu sind die beiden folgenden semantischen Gliederungen S_2 und S_3:

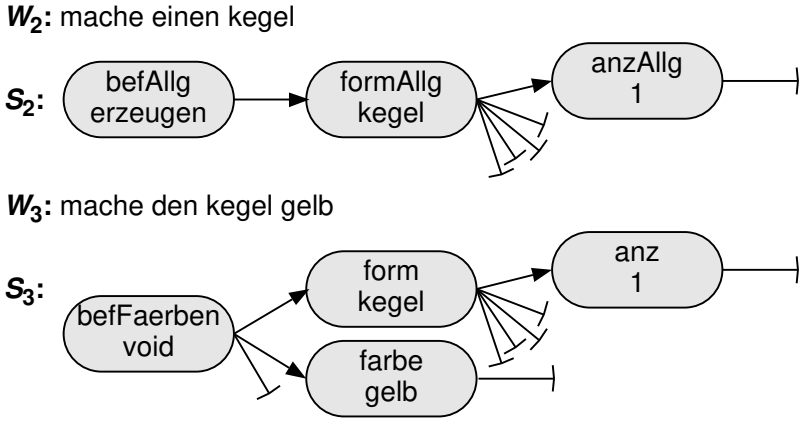


Abb. 4.3: Wortketten W_2 und W_3 mit semantischen Gliederungen S_2 und S_3

Bei demjenigen Semun, welches die Objektform beschreibt, wird eine Typenunterscheidung in „form" und „formAllg" vorgenommen. Durch entsprechende Wahrscheinlichkeiten erster Ordnung im semantischen Modell kann folgendes festgelegt sein:

- Ein Semun vom Typ „befAllg" kann von einem Semun des Typs „formAllg" gefolgt werden, jedoch nicht von einem Semun des Typs „form".
- Ein Semun vom Typ „formAllg" kann von einem Semun des Typs „anzAllg" gefolgt werden, jedoch nicht von einem Semun des Typs „anz".

Damit ist ausgeschlossen, daß der Nachfolger eines Semuns vom Typ „befAllg" von einem Semun des Typs „anz" gefolgt wird. Dies entspricht bereits einer Abhängigkeit 2. Ordnung. Eine ähnliche Abfolgevorschrift läßt sich auch bei „befFaerben" – „form" – „anz" durch geeignete Folgewahrscheinlichkeiten erzwingen.

Der Kontext ist bei der semantischen Decodierung durchaus bedeutungsrelevant. Ein wichtiger Hinweis darauf, daß das Wort „mache" in W_2 ein Erzeugungs-Befehl ist, ist das

unbestimmte Zahlwort „einen“, da in der Regel ein unbestimmtes Objekt nicht zu manipulieren, sondern zu erzeugen ist. Das entsprechende Semun ist jedoch kein unmittelbarer Nachfolger des ‚Erzeugungs‘-Semuns, so daß sich mit bedingten Wahrscheinlichkeiten erster Ordnung die Abfolge „befAllg“ – ... – „anzAllg“ nicht klar erfassen ließe. Dies wird jedoch durch das Einführen des Typs „formAllg“ vermieden.

4.2 Das syntaktische Modell

Das syntaktische Modell ist diejenige Wissenbasis, welche die bedingte Wahrscheinlichkeit $P(W|S)$ für das Auftreten einer Wortkette W gegeben eine semantische Gliederung S bereitstellt. Diese Wahrscheinlichkeit wird im betrachteten ‚top-down‘-Ansatz für den Wortketten-Generator benötigt. Die Vielfalt von W und S ist jedoch zu groß, um $P(W|S)$ direkt aus einem Trainingskorpus zu schätzen, denn es ist nicht anzunehmen, daß alle denkbaren Wortketten und semantischen Gliederungen auch im Training gesehen werden. Das syntaktische Modell muß demnach (wie auch das vorher beschriebene semantische Modell) eine endliche Anzahl Parameter enthalten, die sich einerseits zuverlässig aus begrenztem Trainingsmaterial bestimmen lassen. Andererseits muß es auf die jeweilige Wahrscheinlichkeit $P(W|S)$ einer unbegrenzten Menge an Wortketten W schließen, die zu einer unbegrenzten Menge semantischer Gliederungen S korrespondieren.

4.2.1 Bezug einer Wortkette zur semantischen Gliederung

Zu jeder semantischen Gliederung läßt sich eindeutig ein syntaktisches Netzwerk, ein endlicher, probabilistischer Zustandsautomat mit Zuständen zur Emission von Worten und Übergängen zwischen diesen Zuständen, definieren. Durch Emissionen steuert ein solches Netzwerk die Wortwahl und durch Übergänge die Wortstellung innerhalb der korrespondierenden Wortkette.

Die semantische Gliederung setzt sich baumförmig und damit hierarchisch aus N kleineren Einheiten, den semantischen Untereinheiten (abgekürzt Semunen) s_n, mit $1 \le n \le N$ zusammen. Analog dazu setzt sich das zugehörige syntaktische Netzwerk ebenfalls hierarchisch aus N kleineren Einheiten, den sogenannten *syntaktischen Modulen* (abgekürzt SMen) $A(s_n)$, mit $1 \le n \le N$ zusammen. Jedes Semun s_n steuert dem syntaktischen Modell ein ihm zugeordnetes SM $A(s_n)$ bei.

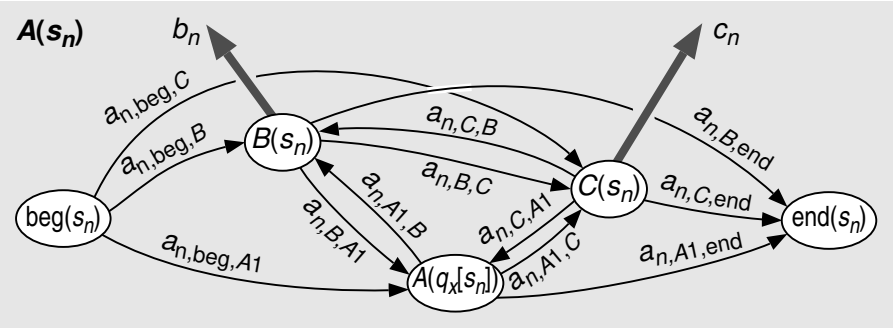


Abb. 4.4: Syntaktisches Modul $A(s_n)$ für ein Semun s_n mit X=1 Nachfolger

Jedes SM $A(s_n)$ steuert die Wortstellung und die Wortwahl der dem Semun s_n zugeordneten Worte. Ein SM ist ein probabilistischer Zustandsautomat, der aus (X+4) Zuständen bzw. Knoten beg(s_n), $A[q_1(s_n)]$, ..., $A[q_X(s_n)]$, $B(s_n)$, $C(s_n)$ und end(s_n) besteht. Dabei markiert X die vom Semun-Typ $t[s_n]$ fest vorgegebene Nachfolger-Anzahl des entsprechenden Semuns s_n.

- Der **Knoten beg(s_n)** wird vom Vorgänger-SM angesprungen. Alle weiteren Zustände des SMs können nur nach beg(s_n) betreten werden. Bei n=1 bildet beg(s_1) den Startpunkt des gesamten syntaktischen Netzwerks.
- Jeder **Zustand $A(q_x[s_n])$** mit $1 \le x \le X$ muß unterscheiden:
  - Falls $q_x[s_n]$ ein Nachfolger-Semun ist, steht dieser Zustand (wie durch dessen Bezeichnung angedeutet) stellvertretend für das SM $A(q_x[s_n])$ des entsprechenden Nachfolger-Semuns $q_x[s_n]$. Das Betreten von $A(q_x[s_n])$ ist gleichbedeutend mit dem Betreten des Startknotens beg($q_x[s_n]$). Sobald der Endknoten end($q_x[s_n]$) erreicht ist, gilt $A(q_x[s_n])$ als abgearbeitet und kann verlassen werden.
  - Falls $q_x[s_n]$ ein leerer Nachfolger ist, gilt $A(q_x[s_n])$ unmittelbar beim Betreten bereits als abgearbeitet und kann verlassen werden, ohne daß irgendein anderes SM berührt wird.
- Der **Zustand $B(s_n)$** emittiert ein bedeutungstragendes Wort $w^+[s_n]$ aus der Wortkette W mit der Emissionswahrscheinlichkeit $b_n = P(\text{Emiss. von } w^+[s_n] \,|\, t[s_n], v[s_n])$.
- Der **Zustand $C(s_n)$** emittiert ein bedeutungsloses Wort $w^-[s_n]$ aus der Wortkette W mit der Emissionswahrscheinlichkeit $c_n = P(\text{Emission von } w^-[s_n] \,|\, t[s_n])$.
- Der **Knoten end(s_n)** beendet die Abarbeitung und springt zu demjenigen Zustand des Vorgänger-SMs, aus dem das aktuelle SM angesprungen wurde. Bei n=1 bildet end(s_1) den Endpunkt des gesamten syntaktischen Netzwerks.

Der Übergang zwischen zwei aufeinanderfolgenden Zuständen wird primär durch die jeweiligen Übergangswahrscheinlichkeiten bestimmt, z.B. markiert die Übergangswahrscheinlichkeit $a_{n,B,C}$ die Wahrscheinlichkeit für einen Übergang vom Zustand $B(s_n)$ zum Zustand $C(s_n)$. Um die in Kap. 3.4 getroffenen Festlegungen einzuhalten, müssen zusätzlich auf dem Pfad von beg(s_n) nach end(s_n) die folgenden Bedingungen zwingend eingehalten werden:

- Keiner der Zustände $A_1(s_n)$, ..., $A_X(s_n)$, $B(s_n)$ oder $C(s_n)$ kann jeweils zweimal betreten werden.
- Der Knoten end(s_n) kann nicht vor den Zuständen $A(q_1[s_n])$, ..., $A(q_X[s_n])$ und $B(s_n)$ betreten werden.

Dadurch wird sichergestellt, daß jeder der Zustände $A(q_1[s_n])$, ..., $A(q_X[s_n])$ und $B(s_n)$ genau einmal und der Zustand $C(s_n)$ maximal einmal betreten wird, was bedeutet, daß jeder Nachfolger genau einmal angesprungen wird sowie genau ein bedeutungstragendes Wort und maximal ein bedeutungsloses Wort emittiert werden. Mit diesen Zusatzbedingungen kann ein SM kein stochastischer Automat erster Ordnung sein, bei dem mögliche Übergänge vom aktuellen in den darauffolgenden Zustand nur vom aktuellen Zustand selbst abhängen.

Ein syntaktisches Netzwerk setzt sich nun aus einer hierarchischen Abfolge solcher SMe zusammen, wobei die Zustände $A(q_X[s_n])$ von den jeweiligen Nachfolger-SMen „aufgefüllt“ sind. Ein konkretes Beispiel für ein syntaktisches Netzwerk einer Äußerung ist in Abb. 4.5 gezeigt.

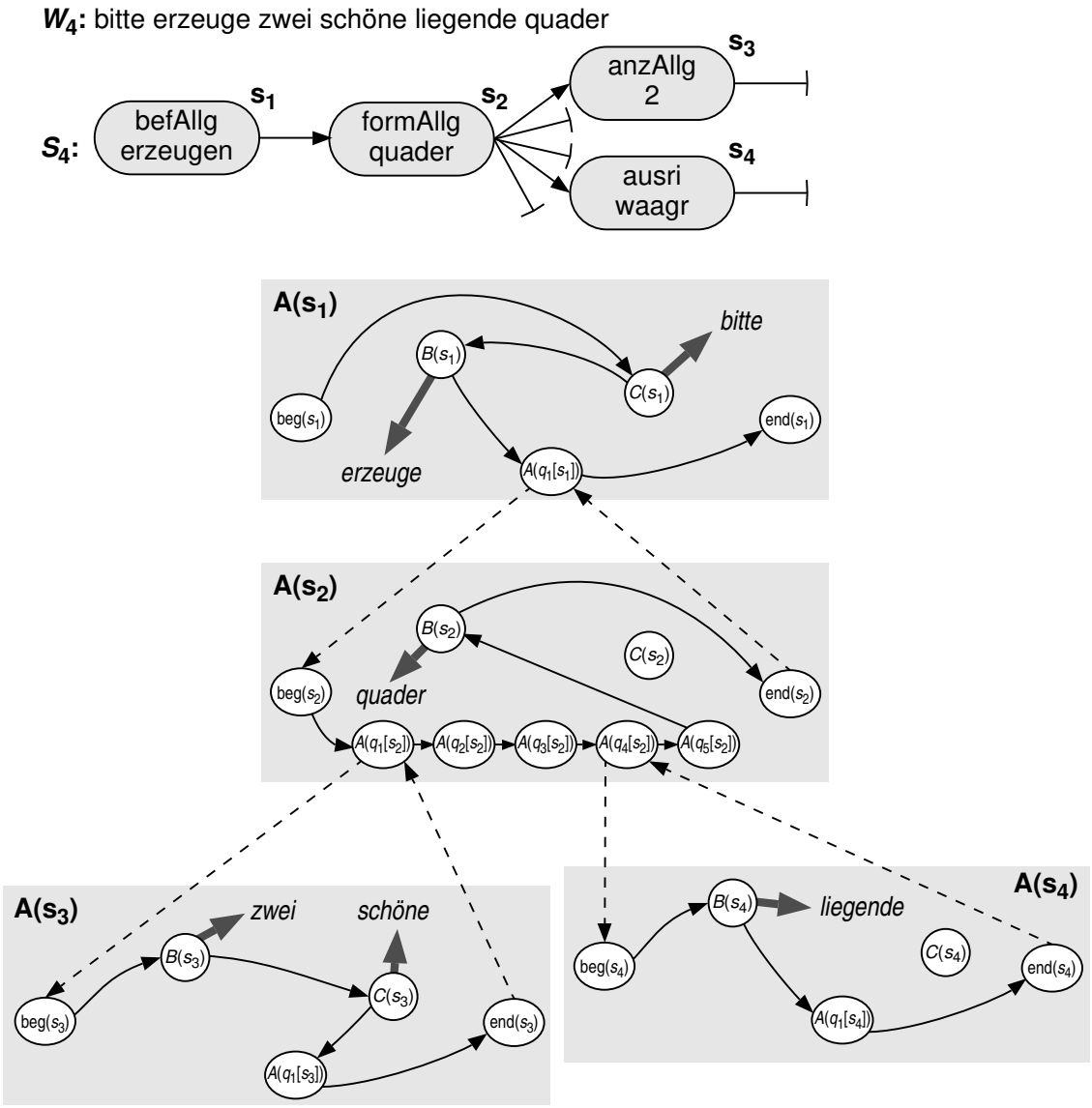


Abb. 4.5: Syntaktisches Netzwerk einer Äußerung mit durchlaufenem Pfad (Darstellung nach [Sta97b])

4.2.2 Wahrscheinlichkeiten im syntaktischen Modell

Vorgeschlagen wird die Abschätzung folgender bedingter Wahrscheinlichkeiten erster Ordnung, aus denen sich die Wahrscheinlichkeit $P(W|S)$ in guter Näherung bestimmen läßt:

- Die **Pfadwahrscheinlichkeit a_n** ist die bedingte Wahrscheinlichkeit, daß in einem zu einem Semun des Typs $t[s_n]$ gehörenden SM genau der vorliegende Pfad durchlaufen wird. Es berechnet sich aus dem Produkt aller Übergangswahrscheinlichkeiten $a_{n,\mu,\nu}$ entlang des Pfades von beg(s_n) nach end(s_n) durch das SM.

$$a_n = P\left(\text{bestimmter Pfad } \underline{\underline{\text{mit max. } P(W|S)}} \,\middle|\, t[s_n]\right) = \prod_{\substack{\text{alle Übergänge} \\ \text{von } \mu \text{ nach } \nu \\ \text{entlang Pfad} \\ \text{durch } A(s_n)}} a_{n,\mu,\nu} \tag{4.7}$$

  Abhängig vom Typ $t[s_n]$ des Semuns s_n muß im syntaktischen Modell eine Matrix von $(X+2)(X+3)$ Übergangswahrscheinlichkeiten $a_{n,\mu,\nu}$ bereitstehen. Somit ist auch die Pfadwahrscheinlichkeit nur vom jeweiligen Semun-Typ abhängig.

  Dabei ist folgendes zu beachten: Falls ein bedeutungsloses Wort $w^-[s_n]$ emittiert und damit der optionale Zustand $C(s_n)$ angesprungen wird, werden auf dem Pfad von beg(s_n) nach end(s_n) $(X+3)$ Übergänge durchlaufen. Wird jedoch kein bedeutungsloses Wort emittiert und $C(s_n)$ ausgelassen, vermindert sich die Anzahl der Übergänge auf $(X+2)$.

  Für einen leeren Nachfolger $q_x[s_n]$ existieren innerhalb des SMs $A(s_n)$ mehrere Pfade, da das entsprechende Nachfolger-Modul $A(q_x[s_n])$ keine weitere Aktion verursacht und deshalb an beliebiger Stelle angesprungen werden kann. In diesem Fall wird derjenige Pfad, welcher zur maximalen Wahrscheinlichkeit $P(W|S)$ führt, bevorzugt. Entsprechendes gilt auch, falls mehrere leere Nachfolger existieren.

- Die **Emissionswahrscheinlichkeit b_n** ist die bedingte Wahrscheinlichkeit, daß der Zustand $B(s_n)$ des SMs, welches zu einem Semun des Typs $t[s_n]$ und des Werts $v[s_n]$ gehört, das zum Semun s_n gehörende bedeutungstragende Wort $w^+[s_n]$ emittiert:

$$b_n = P\Big(\text{Emission von } w^+[s_n] \,\Big|\, t[s_n], v[s_n]\Big) \tag{4.8}$$

  Diese Wahrscheinlichkeit ist vom Typ und vom Wert des zugehörigen Semuns abhängig. Das syntaktische Modell muß für alle Kombinationen aus Typ, Wert und bedeutungstragendem Wort eine solche Emissionswahrscheinlichkeit bereitstellen.

- Die **Emissionswahrscheinlichkeit c_n** ist die bedingte Wahrscheinlichkeit, daß der Zustand $C(s_n)$ des zu einem Semun des Typs $t[s_n]$ gehörenden SMs das bedeutungslose Wort $w^-[s_n]$ emittiert. Zur Vereinfachung von Gl. (4.10) ist c_n immer gleich Eins, falls der durch das SM führende Pfad nicht den optionalen Zustand $C(s_n)$ beinhaltet.

$$c_n = \begin{cases} P\Big(\text{Emission von } w^-[s_n] \,\Big|\, t[s_n]\Big) & , \text{ falls } C(s_n) \text{ betreten} \\ 1 & , \text{ sonst} \end{cases} \tag{4.9}$$

  Falls $C(s_n)$ betreten wird, ist diese Wahrscheinlichkeit nur vom Typ des zugehörigen Semuns abhängig. Das syntaktische Modell muß für alle Kombinationen aus Typ und bedeutungslosem Wort eine solche Emissionswahrscheinlichkeit bereitstellen.

Die **Modulwahrscheinlichkeit d_n** des gesamten SMs $A(s_n)$ läßt sich aus den vorangegangenen Wahrscheinlichkeiten bilden, indem die Pfad- und die beiden Emissionswahrscheinlichkeiten des SMs aufmultipliziert werden:

$$d_n = a_n \cdot b_n \cdot c_n \tag{4.10}$$

4.2.3 Bedingte Wahrscheinlichkeit einer Wortkette

Unter des Annahme der statistischen Unabhängigkeit aller Modulwahrscheinlichkeiten d_n ergibt sich die bedingte Wahrscheinlichkeit $P(W|S)$ als Produkt der Modulwahrscheinlichkeiten aller N vorkommenden SMe des syntaktischen Modells:

$$\boxed{P(W|S) = \prod_{n=1}^{N} d_n = \prod_{n=1}^{N} (a_n \cdot b_n \cdot c_n)} \tag{4.11}$$

Als Beispiel sei im folgenden die Wahrscheinlichkeit $P(W_4|S_4)$ für die in Abb. 4.5 gezeigte Äußerung berechnet:

- Das SM $A(s_1)$ wird beaufschlagt mit $d_1 = a_1 \cdot b_1 \cdot c_1$
  - wobei sich die Pfadwahrscheinlichkeit zu
    $a_1 = a_{1,\,\text{beg},\,C} \cdot a_{1,\,C,\,B} \cdot a_{1,\,B,\,A1} \cdot a_{1,\,A1,\,\text{end}}$
  - und sich die Emissionswahrscheinlichkeiten zu
    $b_1 = P(\text{Emission von } erzeuge | t = \text{befAllg}\,,\, v = \text{erzeugen})$ und
    $c_1 = P(\text{Emission von } bitte | t = \text{befAllg})$ berechnen.
- Das SM $A(s_2)$ wird beaufschlagt mit $d_2 = a_2 \cdot b_2 \cdot c_2$
  - wobei sich die Pfadwahrscheinlichkeit zu
    $a_2 = a_{2,\,\text{beg},\,A1} \cdot a_{2,\,A1,\,A2} \cdot a_{2,\,A2,\,A3} \cdot a_{2,\,A3,\,A4} \cdot a_{2,\,A4,\,A5} \cdot a_{2,\,A5,\,B} \cdot a_{2,\,B,\,\text{end}}$
  - und sich die Emissionswahrscheinlichkeiten zu
    $b_2 = P(\text{Emission von } quader | t = \text{formAllg}\,,\, v = \text{quader})$ und
    $c_2 = 1$ berechnen.
- Das SM $A(s_3)$ wird beaufschlagt mit $d_3 = a_3 \cdot b_3 \cdot c_3$
  - wobei sich die Pfadwahrscheinlichkeit zu
    $a_3 = a_{3,\,\text{beg},\,B} \cdot a_{3,\,B,\,C} \cdot a_{3,\,C,\,A1} \cdot a_{3,\,A1,\,\text{end}}$
  - und sich die Emissionswahrscheinlichkeiten zu
    $b_3 = P(\text{Emission von } zwei | t = \text{anzAllg}\,,\, v = 2)$ und
    $c_3 = P(\text{Emission von } schoene | t = \text{anzAllg})$ berechnen.
- Das SM $A(s_4)$ wird beaufschlagt mit $d_4 = a_4 \cdot b_4 \cdot c_4$
  - wobei sich die Pfadwahrscheinlichkeit zu
    $a_4 = a_{4,\,\text{beg},\,B} \cdot a_{4,\,B,\,A1} \cdot a_{4,\,A1,\,\text{end}}$
  - und sich die Emissionswahrscheinlichkeiten zu
    $b_4 = P(\text{Emission von } liegende | t = \text{ausri}\,,\, v = \text{waagr})$ und
    $c_4 = 1$ berechnen.

$$P(W_4|S_4) = d_1 \cdot d_2 \cdot d_3 \cdot d_4$$

Bei dem vorangegangenen Beispiel ist nur innerhalb des SMs $A(s_1)$ ein einziger Pfad möglich, da im zugehörigen Semun s_1 keine leeren Nachfolger und damit in $A(s_1)$ kein aktionsloser Zustand vorkommt. Die Semune aller übrigen SMe besitzen jedoch leere Nachfolger. Somit sind in diesen SMen auch weitere Pfade, als der jeweils dargestellte, denkbar und möglich. Es wird vorausgesetzt, daß der angegebene Pfad die jeweilige Wahrscheinlichkeit $P(W_4 \big| S_4)$, wie in Gl. (4.7) angedeutet, maximiert.

Kapitel 5

Training der stochastischen Grammatik

Stochastische bzw. probabilistische Wissensbasen beschreiben im allgemeinen Ereignisse und damit direkt oder indirekt verknüpfte Zusammenhänge mittels Wahrscheinlichkeiten. Die Aufgabe des Trainings ist es, mittels zur Verfügung stehender Ereignisse (den sogenannten *Trainingsdaten*) diese Wahrscheinlichkeiten so abzuschätzen, daß nicht nur jene dem Training zur Verfügung stehenden Ereignisse, sondern auch weitere Ereignisse (die sogenannten *Testdaten*, die nicht in der Menge der Trainingsdaten enthalten sein müssen) möglichst genau modelliert werden können. Im Idealfall soll mit einer endlichen Zahl von Trainingsdaten auf eine theoretisch unbegrenzte Menge von Testdaten generalisiert werden.

Im vorliegenden Fall können diese Ereignisse gegebene semantische Gliederungen, Wortketten, Phonemketten oder Beobachtungsfolgen sein, deren Auftreten, deren innere Strukturen und deren Beziehungen zueinander mittels der in den vorausgegangenen Kapiteln festgelegten Wahrscheinlichkeiten beschrieben werden.

5.1 Benötigte Trainingsdaten

Man benötigt für die verschiedenen Wissensbasen jedoch unterschiedliche Trainingsdaten, die dem jeweiligen Trainingsvorgang zur Verfügung gestellt werden müssen. Folgendes Trainingsmaterial wird benötigt:

- Zur Abschätzung der a-priori Wahrscheinlichkeiten $P(S)$ viele semantische Gliederungen,
- zur Abschätzung der bedingten Wahrscheinlichkeiten $P(W|S)$ viele Wortketten, die entsprechenden semantischen Gliederungen sowie die jeweiligen Zusammenhänge zwischen den einzelnen Semunen und den einzelnen Worten,
- zur Abschätzung der bedingten Wahrscheinlichkeiten $P(Ph|W)$ viele Phonemketten mit den entsprechenden Worten und außerdem
- zur Abschätzung der bedingten Wahrscheinlichkeiten $P(O|Ph)$ viele akustische Realisierungen (d.h. die Allophone) mit den entsprechenden Phonemen.

Für die Grafikeditor-Domäne wurden viele gesprochene Äußerungen von verschiedenen Versuchspersonen gesammelt. Folgende Einschränkungen mußten dabei beachtet werden: Jede gesprochene Äußerung

- mußte in deutscher Sprache sein,
- mußte sich innerhalb der „Grafikeditor“-Domäne befinden,
- durfte keinen Nebensatz aufweisen und
- durfte keine spontansprachlichen Effekte[18)] enthalten.

Das akustische Modell kann prinzipiell von bestehenden Spracherkennungssystemen (z.B. SPICOS [Hög90], SPRING [Wot89]) übernommen werden. Dies wäre ohne irgendeine Änderung möglich, falls das betreffende akustische Modell alle erforderlichen Phoneme enthält[19)]. Da innerhalb des hier beschriebenen Systems beim phonetischen Modell keine Aussprachevarianten zulässig sind (d.h. nur eine Phonemkette pro Wort), kann die phonetische Umschrift aus Lexika [Dro62] oder von bereits existierenden phontischen Modellen obiger Spracherkennungssysteme abgeleitet werden. Dabei ist darauf zu achten, daß das jeweils verwendete Phoneminventar mit dem des akustischen Modells identisch sein muß und daß das gewünschte Vokabular darin eingetragen ist.

Aus diesen Gründen müssen nur die Wahrscheinlichkeiten des semantischen und des syntaktischen Modells abgeschätzt (d.h. trainiert) werden. Dazu bedarf es wie oben angesprochen lediglich vieler Wortketten *W*, vieler semantischer Gliederungen *S* und der jeweiligen Verbindungen zwischen *W* und *S*.

5.2 Erstellen des Trainingsmaterials

5.2.1 Sammeln gesprochener Äußerungen mit der Offline-Methode

Dieser einfache Test wurde gemacht, um einen ersten Eindruck über die Art und Weise zu bekommen, wie Benutzer zu einem Computer sprechen. Verschiedene Versuchspersonen sollten einem sprachverstehenden Grafikeditor Kommandos geben, um eine vorgegebene Bildersequenz, die entweder auf Papier gezeichnet oder auf dem Monitor angezeigt wurde, zu verändern. Jedes Bild bestand aus zwei Fenstern, die mit „Fenster vorher“ und „Fenster nachher“ betitelt waren. Diese Fenster sollten die Fenster eines Grafikeditors mit einigen zweidimensionalen Objekten auf dem Monitor repräsentieren.

---

18) Beispiele für spontansprachliche Effekte: Wiederholung oder Auslassung von Worten, Versprecher, Abbrüche, Stottern oder Räuspern. Füllwörter (z.B. „ähm“, „hm“, „oh“) werden in das syntaktische Modell als bedeutungslose Worte aufgenommen.

19) Im Zuge der Systemoptimierung wurden akustische und phonetische Modelle eines am Lehrstuhl entwickelten Spracherkennungssystems implementiert [Beh95][Pla95][Wol96]. Mit diesen Wissensbasen konnte die semantische Erkennungsrate signifikant gesteigert werden [Sta97b].

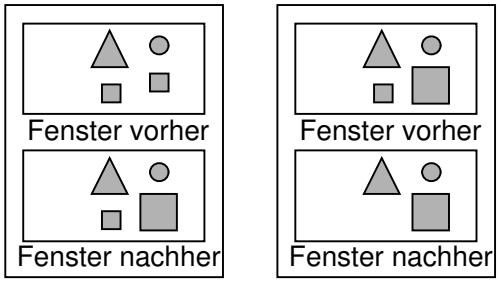


Abb. 5.1: Beispiele zweier Bilder der Offline-Methode

Bei den Versuchspersonen konnte unabhängig von deren technischer Vorbildung beobachtet werden, daß sie vor dem Monitor sitzend eine computerähnliche Sprache bevorzugen. Im Gegensatz dazu sprechen sie meist längere Kommandos, wenn ihnen die Szene auf Papier gezeigt wurde [Mar94].

Wenn die Bilder auf einem Papier dargeboten werden, benötigt die Offline-Methode keinen Computer, sondern nur einen Cassettenrecorder, um die gesprochen Äußerungen zu speichern. Aus diesem Grund kann dieses Experiment nahezu überall durchgeführt werden und ist besonders für solche Personen geeignet, die Unbehagen oder gar Angst davor verspüren, mit einer Maschine zu sprechen. (Dieses Verhalten kann bereits beim Besprechen von Anrufbeantwortern beobachtet werden.)

Der größte Nachteil ist die enorm eingeschränkte Variabilität der Äußerungen (begrenzte Auswahl von Formen, Farben, Größen usw.), da die Szenen und damit auch die den jeweiligen Äußerungen zugrundeliegende Bedeutungsinhalte von vorneherein festgelegt sind. Es ist daher nicht möglich, mit der Offline-Methode semantische Parameter abzuschätzen, so daß sich diese Methode der Datensammlung als unbrauchbar zum Training des semantischen Modells erweist. Es ist aber durchaus möglich, syntaktische Parameter zu gewinnen, in unserem Fall die Wahrscheinlichkeiten des syntaktischen Modells, oder auch Parameter eines (*n*-gramm-) Sprachmodells.

5.2.2 Sammeln gesprochener Äußerungen mit ‚Wizard of Oz'-Simulation

Um eine reale Applikation mit einer unbegrenzten Auswahl semantischer Inhalte innerhalb der Grafikeditor-Domäne zu simulieren, wurde eine ‚Wizard of Oz'-Simulation (WOZ) durchgeführt. Innerhalb einer solchen WOZ simuliert ein menschlicher „Zauberer" (= „Wizard") die sprachverstehende Funktion des Computers, während die Versuchsperson glaubt, mit einer echten Computerapplikation zu kommunizieren. Beschreibungen solcher ‚Wizard of Oz'-Simulationen finden sich u.a. in [Blo93], [Dyb93] oder [Kat94].

In der WOZ der vorliegenden Arbeit hat die Versuchsperson die Aufgabe, eine dreidimensionale Grafik auf dem Monitor nur durch gesprochene Anweisungen zu editieren. Dabei können Objekte wie Kegel, Kugeln, Quader oder Zylinder erzeugt, verändert oder gelöscht werden. Einzige Vorgaben sind, nur deutschsprachige Anweisungen aus der Grafikeditor-Domäne ohne Nebensätze einzugeben. Die Versuchsperson wird in der Regel nicht über den Ablauf des Experimentes in Kenntnis gesetzt und glaubt somit, mit ei-

nem echten sprachverstehenden Programm zu kommunizieren. Aus diesem Grund bietet eine WOZ eine effiziente Möglichkeit, authentische Sprachdaten zu sammeln.

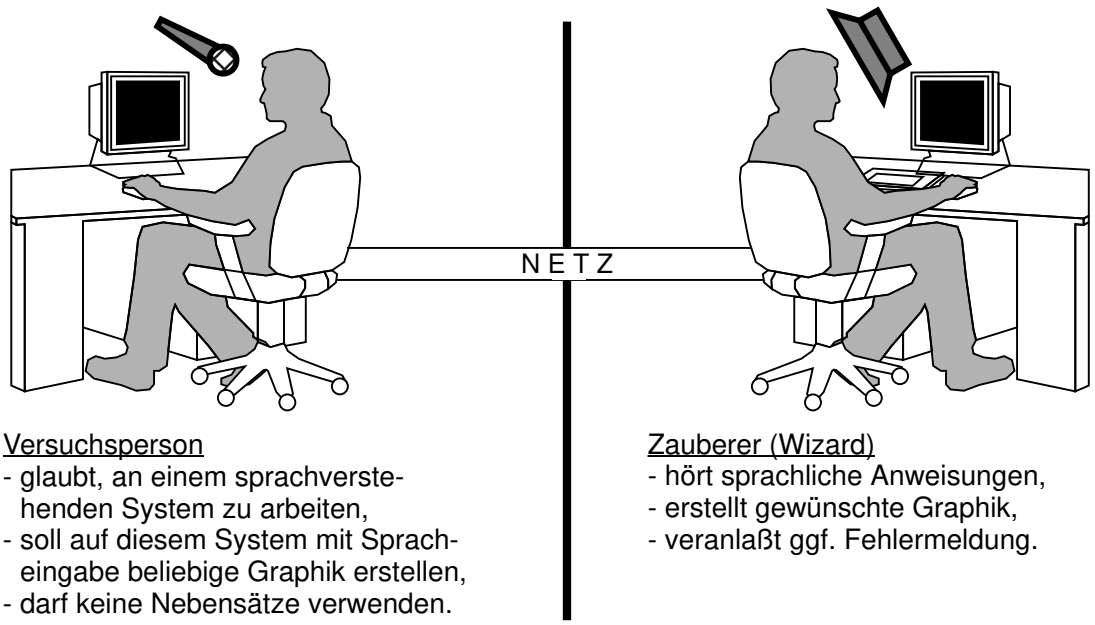


Abb. 5.2: Prinzip der ‚Wizard of Oz'-Simulation

Durch eine WOZ konnten aus der Grafikeditor-Domäne von 33 Versuchspersonen insgesamt 1915 gesprochene Äußerungen in deutscher Sprache gesammelt werden. Davon waren 1843 Äußerungen im Sinne der Vorgaben korrekt, d.h. innerhalb der Domäne ‚Grafikeditor', ohne Nebensatz sowie ohne spontansprachliche Effekte (siehe Fußnote 18 auf Seite 56). Das sich ergebende Vokabular resultierte in 853 Worteinträgen [Mül95c].

5.2.3 Sammeln geschriebener Äußerungen über das WWW

Als Demoapplikation wurde der natürlichsprachliche Grafikeditor NASGRA, als interaktive WWW[20)]-Seite implementiert. Die dabei verwendeten Wissensbasen (semantisches und syntaktisches Modell) wurden mit den 1843 Äußerungen der WOZ trainiert. Die Adresse, mit der weltweit auf dieses Programm zugegriffen werden kann, lautet:

http://www.mmk.e-technik.tu-muenchen.de/~mue/nasgra/

Diese Applikation erlaubt deutschsprachige, textuelle (d.h. ‚getippte') Eingabe von Äußerungen aus der Grafikeditor-Domäne. Da sämtliche Äußerungen aufgezeichnet werden, ist dies eine ideale Methode, um ohne zeitlichen Aufwand eine große Anzahl von domänenspezifischen Wortketten zu sammeln.

Da diese Anwendung noch läuft und bislang noch nicht quantitativ evaluiert wurde, liegen im Rahmen der vorliegenden Arbeit noch keinerlei Ergebnisse dieser Trainingsdaten-

20) WWW bedeutet *World Wide Web* (auf Deutsch *weltweites Spinnennetz*) und bezeichnet ein über das Internet zugängliches dezentrales Informationssystem aus beliebig vielen miteinander vernetzten Rechnern.

gewinnung vor. Ein grober Überblick über die bislang gesammelten Wortketten bestätigt die Vermutung, daß eine geschriebene Äußerung wesentlich weniger bedeutungslose Worte enthält als eine gesprochene.

5.2.4 Erstellen von Wortketten und semantischen Gliederungen

Als unmittelbare Eingabe zum Training des semantischen und des syntaktischen Modells wird zu jeder Äußerung manuell eine ASCII-Datei erstellt. Diese Datei enthält neben der bereits vorhandenen oder noch zu schreibenden Wortkette $W = w_1 w_2 \ldots w_{nw} \ldots w_{NW}$, die korrespondierende semantische Gliederung $S = \{s_1, s_2, \ldots, s_n, \ldots, s_N\}$ und eine exakte Zuordnung

$$nw \rightarrow n \tag{5.1}$$

der Zugehörigkeit jedes Wortes w_{nw} aus W zu einem Semun $s_n \in S$.

Die semantischen Gliederungen der ersten etwa 750 Äußerungen mußten jeweils rein manuell erstellt werden. Mit den daraus erstellten semantisch-syntaktischen Modellen konnte im weiteren Verlauf ein großer Teil der nachfolgenden Wortketten automatisch in korrespondierende semantische Gliederungen gewandelt werden. Um jedoch sicher zu gehen, daß die semantische Decodierung in jedem Falle korrekt durchgeführt wurde, mußten sämtliche Dateien noch einmal überprüft werden. Die Erstellung dieser Dateien ist wohl der zeitintensivste Prozeß innerhalb des Trainings und muß für neues Trainingsmaterial (z.B. für eine neue Domäne, bei Erweiterung des Wortschatzes oder für eine neue Sprache) aufs neue durchlaufen werden.

5.3 Bestimmung der Wahrscheinlichkeiten des semantischen Modells

Die Wahrscheinlichkeiten im semantischen Modell werden zunächst initialisiert und daraufhin gegebenenfalls mehrfach iterativ optimiert. Um weitere sinnvolle, aber nicht im Trainingsmaterial vorkommende Nachfolgerscharen zuzulassen, werden ‚Floor-Value'-Folgewahrscheinlichkeiten eingeführt.

5.3.1 Initialisierung

Die entsprechenden Wurzel-, Wert- und Folgewahrscheinlichkeiten des vorliegenden Trainingsmaterials werden bestimmt, indem aus allen jeweils auftretenden Häufigkeiten durch Verhältnisbildung die entsprechenden Wahrscheinlichkeiten abgeleitet werden.

- Für die **Wurzelwahrscheinlichkeit** $P(\tau)$ eines Typs τ ergibt sich:

$$P(\tau) = \frac{\text{Anzahl aller Wurzel-Semune mit Typ } \tau}{\text{Anzahl aller semantischer Gliederungen}} \tag{5.2}$$

Das semantische Modell muß die Wurzelwahrscheinlichkeiten $P(\tau)$ aller im Trainingsmaterial vorkommenden Typen τ mit $\tau \neq$ leer enthalten.

- Für die **Wertwahrscheinlichkeit** $P(\varphi|\tau)$ eines zum Typ τ gehörenden Werts φ ergibt sich:

$$P(\varphi|\tau) = \frac{\text{Anzahl aller Semune mit Typ } \tau \text{ und Wert } \varphi}{\text{Anzahl aller Semune mit Typ } \tau} \tag{5.3}$$

  Das semantische Modell muß die Wertwahrscheinlichkeiten $P(\varphi|\tau)$ aller im Trainingsmaterial vorkommenden Kombinationen aus einem Typ τ mit $\tau \neq$ leer und einem Wert φ enthalten.

- Für die **Folgewahrscheinlichkeit** $P(\tau_1, \tau_2, \ldots, \tau_X|\tau)$ einer zum Typ τ gehörenden Nachfolgerschar $(\tau_1, \tau_2, ..., \tau_X)$ ergibt sich:

$$P(\tau_1, \tau_2, \ldots, \tau_X|\tau) = \frac{\text{Anz. aller Semune mit Typ } \tau \text{ u. Nachf.schar } \tau_1, \tau_2, \ldots, \tau_X}{\text{Anzahl aller Semune mit Typ } \tau} \tag{5.4}$$

  Das semantische Modell muß die Folgewahrscheinlichkeiten $P(\tau_1, \tau_2, \ldots, \tau_X|\tau)$ aller im Trainingsmaterial vorkommenden Kombinationen aus den Typen τ, τ_1, τ_2, ..., τ_X enthalten. Dabei ist zu beachten, daß ein Typ oder mehrere Typen τ_1, τ_2, ..., τ_X auch ‚leer' sein können. Für den Typ τ gilt immer $\tau \neq$ leer .

Mit allen diesen Wurzel-, Wert- und Folgewahrscheinlichkeiten setzt sich das initialisierte semantische Modell zusammen.

5.3.2 Iteration

Das nun folgende Training ist ein iterativer Prozeß, der sich solange wiederholt, bis sich das daraus resultierende iterierte semantische Modell mit demjenigen Modell des vorangegangenen Initialisierungs- bzw. Iterationsschrittes identisch ist. Die Iteration wird mindestens einmal durchgeführt.

Grundsätzlich wird bei der iterativen Optimierung des semantischen Modells die im Kapitel 3.3 beschriebene Eigenschaft der semantischen Gliederung ausgenützt, daß sich Semune innerhalb einer Linie mit jeweils einem Nachfolger beliebig austauschen lassen. In allen semantischen Gliederungen des Trainingsmaterials werden zu jeder vorkommenden aus m Semunen bestehenden Linie sämtliche $m!$ kombinatorische Anordnungen gebildet. Diejenige Linie mit maximalem Produkt P_{opt} aus den darin enthaltenen Folgewahrscheinlichkeiten und den Folgewahrscheinlichkeiten zu den Nachbar-Semunen (die Wahrscheinlichkeiten werden aus dem Modell des vorangegangenen Initialisierungs- bzw. Iterationsschrittes bezogen) wird anstatt der ursprünglichen Linie in die vorliegende semantische Gliederung eingesetzt. Dabei muß jedoch unbedingt sichergestellt sein, daß der in Kap. 3.4 festgelegte Bezug zwischen einer Wortkette und der korrespondierenden semantischen Gliederung aufrechterhalten wird. Damit kann das semantische Modell erneut erstellt werden, indem alle jeweils auftretenden Häufigkeiten wie im vorausgegan-

genen Kapitel 5.3.1 bestimmt und die entsprechenden Verhältnisse gebildet werden [Kai95a][Kai95b].

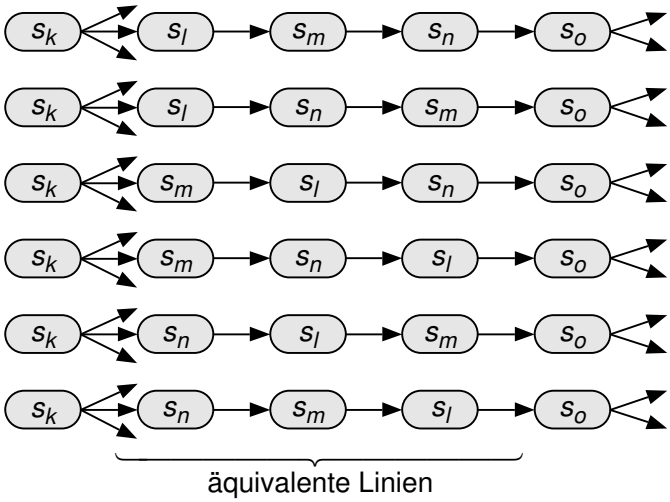


Abb. 5.3: Beispiel für die wahlfreie Anordnung von Semunen innerhalb der Linie $\binom{l}{n}$ von Semunen mit jeweils einem Nachfolger. (Da die Semune s_k und s_o mehr als einen Nachfolger besitzen, ist deren Anordnung fest.) Jede Linie beinhaltet dieselbe Information; damit gelten alle daraus gebildete semantische Gliederungen als äquivalent.

In Abb. 5.3 ist die Vertauschbarkeit von Semunlinien exemplarisch dargestellt. Das Produkt P_{opt} aus den jeweils betroffenen Folgewahrscheinlichkeiten würde sich dabei wie folgt berechnen:

$$
\begin{aligned}
P_{\text{opt}} \;=\; \max\,[\;
& P(\ldots, s_l, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_m] \,|\, t\,[s_l]) \cdot P(t\,[s_n] \,|\, t\,[s_m]) \cdot P(t\,[s_o] \,|\, t\,[s_n]) \;, \\
& P(\ldots, s_l, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_n] \,|\, t\,[s_l]) \cdot P(t\,[s_m] \,|\, t\,[s_n]) \cdot P(t\,[s_o] \,|\, t\,[s_m]) \;, \\
& P(\ldots, s_m, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_l] \,|\, t\,[s_m]) \cdot P(t\,[s_n] \,|\, t\,[s_l]) \cdot P(t\,[s_o] \,|\, t\,[s_n]) \;, \\
& P(\ldots, s_m, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_n] \,|\, t\,[s_m]) \cdot P(t\,[s_l] \,|\, t\,[s_n]) \cdot P(t\,[s_o] \,|\, t\,[s_l]) \;, \\
& P(\ldots, s_n, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_l] \,|\, t\,[s_n]) \cdot P(t\,[s_m] \,|\, t\,[s_l]) \cdot P(t\,[s_o] \,|\, t\,[s_m]) \;, \\
& P(\ldots, s_n, \ldots \,|\, t\,[s_k]) \cdot P(t\,[s_m] \,|\, t\,[s_n]) \cdot P(t\,[s_l] \,|\, t\,[s_m]) \cdot P(t\,[s_o] \,|\, t\,[s_l]) \;]
\end{aligned}
\tag{5.5}
$$

Da die semantischen Gliederungen des Trainingsmaterials gegebenenfalls bezüglich P_{opt} umgestellt werden, wirkt sich die iterative Optimierung nicht nur auf das semantische Modell, sondern auch auf das Trainingsmaterial selbst aus. Damit lassen sich schnell und effektiv Inkonsistenzen beseitigen, die bei der manuellen Aufstellung der semantischen Gliederungen fälschlicherweise aufgetreten sein könnten.

5.3.3 ‚Floor-Value'-Folgewahrscheinlichkeiten

Im Idealfall wäre das Trainingsmaterial groß genug, um alle denkbaren und sinnvollen Nachfolgerscharen zu enthalten. In der Realität sind jedoch nur ein Teil davon im Trai-

ningsmaterial erfaßt. Daher kann es durchaus sein, daß ein bestimmtes Semun vorliegt, von dem nicht alle möglichen Nachfolgerscharen im Trainingsmaterial vorkommen. Sollte jedoch eine solche unerfaßte Nachfolgerschar in einem zu decodierenden Testsatz vorkommen, könnte die vorliegende semantische Gliederung nicht oder nur fehlerhaft verarbeitet werden. Um das semantische Modell in die Lage zu versetzen, möglichst viele sinnvolle Nachfolgerscharen zu generieren, werden dem Modell sogenannte *‚Floor Value'-Wahrscheinlichkeiten* (auf Deutsch „Bodensatz-Wahrscheinlichkeiten") hinzugefügt, denen die niedrigste bereits vorkommende Folgewahrscheinlichkeit zugeordnet wird. Anschließend werden alle Folgewahrscheinlichkeiten des betrachteten Typs normiert, so daß die Summe über sie den Wert Eins ergibt.

Als Beispiel betrachten wir im folgenden einen Typ τ mit X=3 Nachfolgern. Im Training ergeben sich für diesen Typ die in Gl. (5.6) aufgeführten Folgewahrscheinlichkeiten:

$$\begin{aligned} P(\tau_1, \tau_2, \text{leer}|\tau) &= 0,6 \\ P(\tau_3, \tau_4, \tau_5\big|\tau) &= 0,3 \\ P(\tau_3, \text{leer}, \text{leer}|\tau) &= 0,1 \end{aligned} \tag{5.6}$$

Dabei können folgende Feststellungen getroffen werden:

- Für den Typ $t\,[q_1]$ des ersten Nachfolgers werden die Typen τ_1, τ_3,
- für den Typ $t\,[q_2]$ des zweiten Nachfolgers werden die Typen τ_2, τ_4, ‚leer' sowie
- für den Typ $t\,[q_3]$ des dritten Nachfolgers werden die Typen τ_5 und ‚leer'

beobachtet. Kombiniert man die vorkommenden Typen unter der Bedingung, daß sich die jeweilige Stellenanordnung nicht ändert, ergeben sich $2 \cdot 3 \cdot 2 = 12$ Anordnungsmöglichkeiten. Da nur die drei in (5.6) aufgeführten Wahrscheinlichkeiten durch das Trainingsmaterial abgedeckt sind, werden nun als „Floor-Values" die neun fehlenden Folgewahrscheinlichkeiten dem semantischen Modell hinzugefügt: $P(\tau_1, \tau_4, \tau_5\big|\tau)$, $P(\tau_1, \text{leer}, \tau_5\big|\tau)$, $P(\tau_1, \text{leer}, \text{leer}|\tau)$, $P(\tau_3, \tau_4, \text{leer}|\tau)$, usw. Alle diese zusätzlich hinzugekommenen Folgewahrscheinlichkeiten werden auf die niedrigste bereits vorkommende Folgewahrscheinlichkeit gesetzt – im vorliegenden Fall wäre dies 0,1. Da neun Wahrscheinlichkeiten mit dem Wert 0,1 hinzukommen, beträgt die Summe über alle Folgewahrscheinlichkeiten des Typs τ zunächst 1,9, was eine notwendige Normierung auf die Gesamtsumme Eins zur Folge hat. Somit ergeben sich als endgültige Folgewahrscheinlichkeiten:

$$\begin{aligned} P(\tau_1, \tau_2, \text{leer}|\tau) &= 0,6/1,9 = 0,3158 \\ P(\tau_3, \tau_4, \tau_5\big|\tau) &= 0,3/1,9 = 0,1579 \\ P(\tau_3, \text{leer}, \text{leer}|\tau) &= 0,1/1,9 = 0,0526 \end{aligned} \tag{5.7}$$

Alle neun hinzukommenden „Floor-Value"-Folgewahrscheinlichkeiten haben die gleiche Wahrscheinlichkeit $0,1/1,9 = 0,0526$:

$$
\begin{aligned}
P(\tau_1, \tau_4, \tau_5 \mid \tau) &= 0,0526 \\
P(\tau_1, \text{leer}, \tau_5 \mid \tau) &= 0,0526 \\
P(\tau_1, \text{leer}, \text{leer} \mid \tau) &= 0,0526 \\
P(\tau_3, \tau_4, \text{leer} \mid \tau) &= 0,0526 \\
&\text{usw}\ldots
\end{aligned}
\tag{5.8}
$$

Sicherlich verschlechtern die „Floor-Value"-Folgewahrscheinlichkeiten das Erkennungsergebnis, wenn Trainings- und Testmaterial identisch sind, da ja nun semantische Gliederungen gebildet werden können, die im Trainingsmaterial nicht vorkommen. Die vorgestellte Methode bewährt sich jedoch, wenn Trainings- und Testmaterial verschieden sind, und führt zu einer Verbesserung der Erkennungsergebnisse [Kai95b]. Letztere Voraussetzung ist bei einem interaktiven Einsatz aber immer gegeben, da von einem Benutzer nicht das Nachsprechen der im Trainingsmaterial vorkommenden Sätze erwartet werden kann. Die Mächtigkeit des semantischen Modells steigt dadurch ungemein, ohne jedoch die sinnvollen Restriktionen der semantischen Gliederung zu verletzen.

5.4 Bestimmung der Wahrscheinlichkeiten des syntaktischen Modells

Die Abschätzung der Übergangs- und Emissionswahrscheinlichkeiten innerhalb des syntaktischen Modells wird ebenfalls als iterativer Prozeß ausgeführt. Dazu werden eine Vielzahl an semantischen Gliederungen, Wortketten und die in Gl. (5.1) aufgezeigte Zuordnung der Worte zu den korrespondierenden Semunen benötigt. Obwohl zu einer unendlichen Zahl denkbarer semantischer Gliederungen auch unendlich viele Wortketten möglich erscheinen, hält sich die Anzahl abzuschätzender Parameter in engen Grenzen, da alle Übergangswahrscheinlichkeiten und die Emissionswahrscheinlichkeiten der bedeutungslosen Worte nur vom Typ eines Semuns abhängen. Lediglich die Emissionswahrscheinlichkeiten der bedeutungstragenden Worte hängen sowohl vom Typ als auch vom jeweiligen Wert ab.

Da die Implementierung keinen Bestandteil der vorliegenden Arbeit darstellt, sei auf eine Darstellung an dieser Stelle verzichtet und auf die ausführliche Beschreibung in [Sta97b] verwiesen.

5.5 Beobachtungen innerhalb des Trainingsmaterials

Für das Design eines sprachverstehenden Systems kann es wichtig sein, etwas über Informationsmenge, Informationskomplexität, Wortanzahl und zeitliche Dauer der Äußerungen zu wissen. Durch eine Auswertung des Trainingsmaterials können quantitative Aussagen über die Beobachtungsfolge O (welche die Eigenschaften des Sprachsignals repräsentiert), die Wortkette W und die entsprechende semantische Gliederung S gemacht werden:

- Eine **Beobachtungsfolge** *O* kann durch ihre zeitliche Dauer T_O charakterisiert werden.
- Eine **Wortkette** *W* kann durch die Anzahl von bedeutungstragenden und bedeutungslosen Worten beschrieben werden.
- Eine **semantische Gliederung** *S* kann durch folgende Werte charakterisiert werden: Die Semun-Anzahl *N* (die mit der Anzahl an bedeutungstragenden Worten identisch ist) repräsentiert die *Informationsquantität*. Die Schachtelungstiefe *D* (das ist die Semun-Anzahl entlang des längsten Pfades innerhalb *S*) zeigt, wie detailliert eine Äußerung ist, und steht demnach für die *Informationskomplexität*.

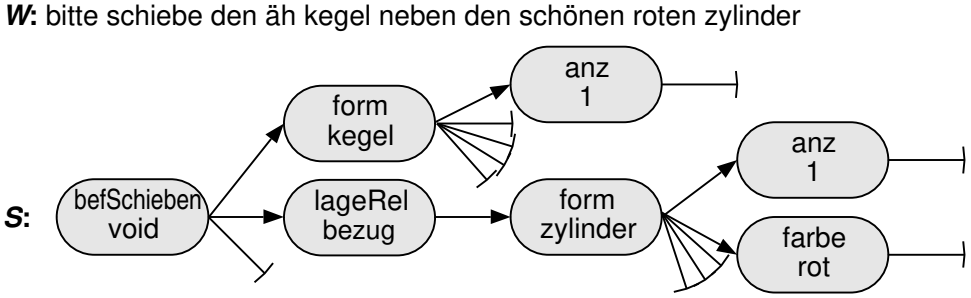


Abb. 5.4: Wortkette *W* und semantische Gliederung *S*

Als Beispiel zeigt obige Abb. 5.4 die Wortkette und die semantische Gliederung der Äußerung „bitte schiebe den äh kegel neben den schönen roten zylinder“, woraus folgende Werte abgeleitet werden können:

- Die Gesamtzahl aller Worte ist 10.
- Die Anzahl der bedeutungslosen Worte ist 3: „bitte“, „äh“ und „schönen“.
- Die Anzahl der bedeutungstragenden Worte, die mit der Semun-Anzahl *N* identisch ist, beträgt 7.
- Die Schachtelungstiefe *D* ist 4.

Zu jeder WOZ wurden alle Versuchspersonen – weiblich (w) oder männlich (m) – gefragt, wie oft sie einen Computer benutzen: intensiv (i), gelegentlich (g) oder nie (n). Werden nun die Beobachtungsfolgen, Wortketten und semantischen Gliederungen all derjenigen, während der WOZ gesammelten Äußerungen ausgewertet, können die folgenden interessanten Beobachtungen gemacht werden [Mül95c]:

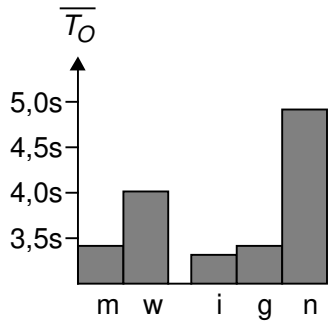


Abb. 5.5: Durchschnittliche Zeitdauer $\overline{T_O}$ pro Äußerung

Abb. 5.5 zeigt die durchschnittliche Zeitdauer $\overline{T_O}$ der gesprochenen Anweisungen. Es kann beobachtet werden, daß männliche Versuchspersonen kürzere Anweisungen sprechen als weibliche. Außerdem steigt die mittlere Zeitdauer von (i)- zu (n)-Versuchspersonen an. Auffällig ist hierbei insbesondere, daß (n)-Versuchspersonen ($\overline{T_O} = 4,92$ s) im Mittel eineinhalbmal so lange brauchen wie (i)-Versuchspersonen ($\overline{T_O} = 3,27$ s), um eine Anweisung zu sprechen.

Eine Äußerung zählt genau dann als gültig, wenn sie sich innerhalb der ‚Grafikeditor'-Domäne befindet, in deutscher Sprache ist, keinen Nebensatz und keine spontansprachliche Effekte beinhaltet. Gemäß Abb. 5.6 zeigte sich, daß (w)- und (g)-Versuchspersonen gerne an die Grenzen des Systems gehen, während (m)-, (i)- und (n)-Versuchspersonen eher die vorgegebenen Einschränkungen respektieren. Manchmal sprachen einige (w)-, (g)- oder (n)-Versuchspersonen Kommentare wie „mir ist langweilig", „hmm was soll ich jetzt tun" oder „oh du bist ein toller rechner", die einem sprachverstehenden System natürlich erhebliche Probleme bereiten können.

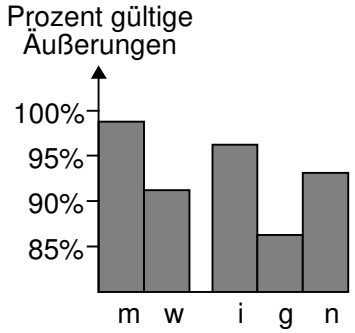


Abb. 5.6: Prozentsatz an gültigen Äußerungen

Es konnte beobachtet werden, daß (i)-Versuchspersonen, die an Computer gewohnt sind und wissen, daß der übliche Mensch-Maschine-Dialog wegen festgelegter Regeln ziemlich unflexibel ist, computerähnliche, sehr kurze Floskeln mit einem Minimum an bedeutungstragenden Worten sowie meist keine bedeutungslose Worte benutzen. Andererseits benutzen (g)- und (n)-Versuchspersonen mehr bedeutungslose Worte innerhalb längerer Äußerungen, die in Extremfällen denjenigen eines Mensch-Mensch-Dialogs ziemlich ähnlich sind. Abb. 5.7 zeigt, daß im Vergleich zu (i)-Versuchspersonen die (g)- und (n)-Versuchspersonen Anweisungen mit mehr Worten und mehr bedeutungslosen Worten ge-

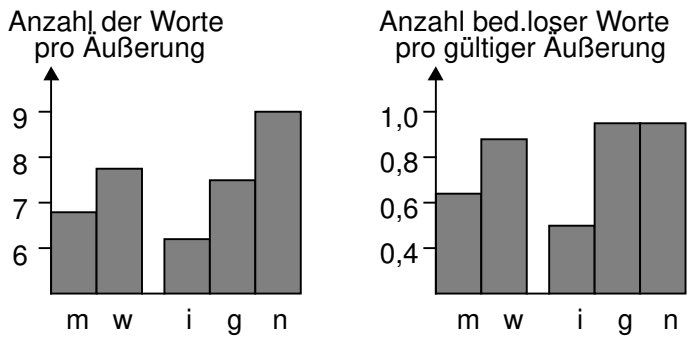


Abb. 5.7: Mittlere Anzahl der Worte pro Äußerung und mittlere Anzahl bedeutungsloser Worte pro gültiger Äußerung

ben. (Die mittlere Anzahl der bedeutungstragenden Worte ist mit der mittleren Semun-Anzahl $\overline{N}$ identisch und ist in Abb. 5.8 aufgeführt.) Eine ähnliche Aussage kann beim Vergleich von männlichen und weiblichen Versuchspersonen getroffen werden.

Abb. 5.8 zeigt, daß Informationsquantität (mittlere Semun-Anzahl $\overline{N}$) und Informationskomplexität (mittlere Schachtelungstiefe $\overline{D}$) von (n)- nach (i)-Versuchspersonen abnimmt. Da die letztere Gruppe in der Regel die Probleme beim automatischen Sprachverstehen kennt, versucht sie, semantische Komplexität ebenfalls zu vermeiden. Erstere Gruppe jedoch ist bezüglich dessen, was der Computer kann, ziemlich unbedarft und vermutet, daß alles, was ein Mensch interpretieren kann, auch für eine Maschine erkennbar ist.

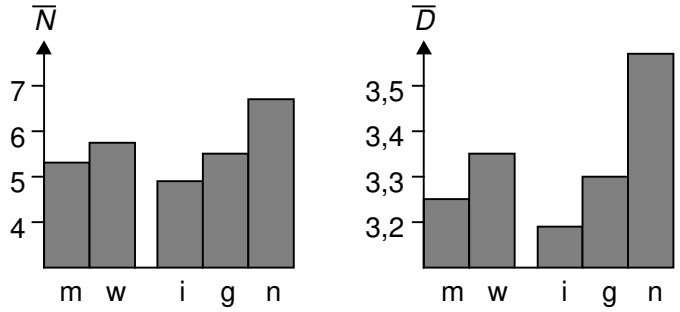


Abb. 5.8: Mittlere Anzahl $\bar{N}$ der Semune und mittlere Schachtelungstiefe $\bar{D}$ pro gültiger Äußerung

In diesem Zusammenhang ist es von Bedeutung, die im gesamten Trainingsmaterial vorkommenden Maximalwerte zu bestimmen:

- Die längste Zeitdauer einer gesprochenen Äußerung beträgt $T_{O,\,\max} = 16,9$ s.
- Die längste Wortkette enthält 25 Worte.
- Die größte Anzahl bedeutungsloser Worte in einer gültigen Wortkette ist 5.
- Die größte Anzahl bedeutungstragender Worte in einer gültigen Wortkette, identisch mit der maximalen Semun-Anzahl ergibt $N_{\max} = 18$.
- Die maximale Schachtelungstiefe einer semantischen Gliederung ist $D_{\max} = 6$.

Angesichts der nahezu unbegrenzten Art und Weise, wie Menschen untereinander Information austauschen, erscheint selbst in einer eng umgrenzten Domäne eine völlig restriktionslose Spracheingabe als eine sehr anspruchsvolle Aufgabe. Die in diesem Kapitel angestellten Beobachtungen zeigen, daß sich die Variabilität natürlicher Sprache auf akustischer, phonetischer, syntaktischer und auch auf semantischer Ebene auswirkt.

Kapitel 6

Intentionsdecoder und Grafikerzeugung

Da die semantische Gliederung als wortnahe Darstellung einer Äußerung stark kontextbehaftet ist und die Umgebungskonstellation sowie vorherige Äußerungen nicht miteinbezieht, eignet sich diese Art der Repräsentation nicht zur unmittelbaren Steuerung einer laufenden Applikation. Vielmehr muß eine nachfolgende Instanz dasjenige Wissen liefern, welches in der semantischen Gliederung nicht zwingend vorhanden ist [Sta97a].

Als triviales Beispiel seien die Wortketten „mache ihn größer“ bzw. „größer machen“ betrachtet. Das Wissen, auf welches Objekt sich die Anweisung bezieht, ist in der semantischen Gliederung nicht vorhanden. Weiterhin kann es bei der Äußerung „lösche den roten quader“ zu Problemen kommen, falls überhaupt kein Quader existiert oder falls mehrere Quader in der Grafik vorhanden sind. Im letzteren Fall müßte der Quader mit weiteren Eigenschaften (z.B. Farbe, Größe usw.) näher spezifiziert werden [Kor96], ansonsten liegt eine Fehleingabe vor. Der Intentionsdecoder verbindet nun eine semantische Gliederung mit der aktuellen Umgebungskonstellation und erzeugt daraus eine lineare Abfolge von maschineninterprtierbaren Befehlen, die im Rahmen dieser Arbeit als Intention *I* bezeichnet werden.

6.1 Allgemeiner Lösungsansatz

Für die Umsetzung einer semantischen Gliederung in einen den Wünschen des Benutzers entsprechenden Datenbankzugriff [21)] wird eine Kombination aus **Präprozessor** (vorverarbeitendes Modul) und **Compiler** (übersetzendes Modul) vorgeschlagen. Im Falle des Grafikeditors realisiert ein nachfolgender **Interpreter** (ausführendes Modul) die vom Compiler ausgegebenen Datenbankzugriffe. Ein ähnlicher Aufbau, bestehend aus einer sequentiellen Abfolge aus Übersetzungs- und Interpretationsphase, hatte sich bereits innerhalb eines Systems zum Verstehen natürlicher Sprache bewährt [Hab80]. Abweichend

21) Innerhalb des implementierten Grafikeditors werden alle vorhandenen Objekte und deren Eigenschaften in einer Datenbank, der sogenannten *Grafikdatenbasis*, gespeichert. Somit liegt im Prinzip jeder Grafikänderung programmtechnisch ein Datenbankzugriff zugrunde.

von den Darstellungen in [Ebe95] und [Ebe96] wird der Interpreter im Rahmen der vorliegenden Arbeit jedoch aus Konsistenzgründen[22)] zum nachfolgenden Systemblock ‚Grafikerzeugung' gezählt.

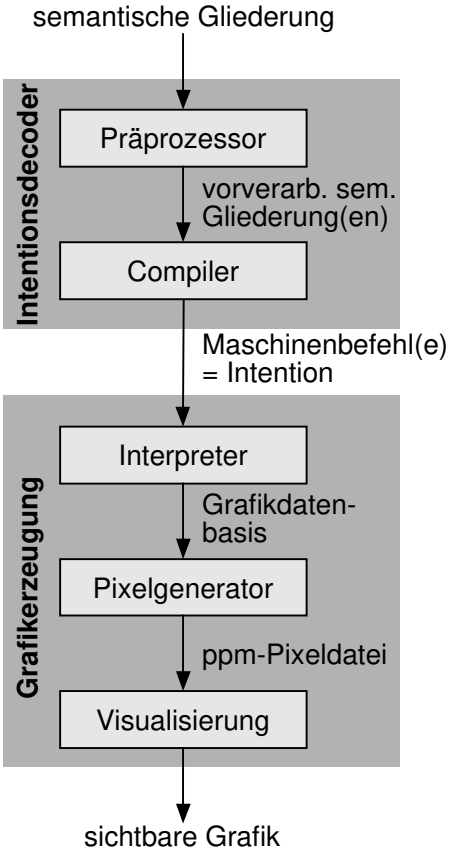


Abb. 6.1: Blockdiagramm des Intentionsdecoders und der Grafikerzeugung

Im hier betrachteten Fall werden die mittels eines Präprozessors aufbereitete semantischen Gliederungen als Quellsprache für den Compiler aufgefaßt. Ziel der Compilierung ist eine Zwischensprache, eine eigens für diesen Zweck entworfene und optimierte Datenbankzugriffssprache zum Manipulieren der Grafikdatenbasis [Ebe95]. (Prinzipiell wäre jedoch auch eine standardisierte Zugriffssprache, z.B. SQL [Pet90], denkbar.) Diese Zugriffssprache ist mit universell einsetzbaren, sogenannten *Elementarbefehlen* ausgestattet, wie Befehle für arithmetische Operationen, Entscheidungen für die Regelung des Datenflusses und Befehle für die Zugriffe auf eine Datenbank.

Vor Inbetriebnahme des Systems müssen sämtliche denkbaren und sinnvollen Semune innerhalb der betrachteten Domäne mit Befehlen der Zwischensprache umschrieben werden. Das Semun und die korrespondierende Befehlsfolge repräsentieren die jeweils gleiche Information. So ist es möglich, auf einfache Weise neue Semune zu integrieren, um der Forderung nach Portabilität auf andere Domänen gerecht zu werden.

22) Bei der Portierung auf die ‚Serviceroboter'-Domäne zeigte sich, daß eine Kombination aus Präprozessor und Compiler zur Intentionsdecodierung ausreicht. Die Schnittstelle der Maschinenbefehle (d.h. Intention) besteht auch bei Applikationen ohne Datenbankkontrolle.

6.2 Präprozessor

Bevor die Abarbeitung einer semantischen Gliederung auf dem Compiler beginnen kann, müssen noch Optimierungen und Umstrukturierungsmaßnahmen durchgeführt werden, um den nachfolgenden Stufen die Arbeit zu erleichtern. Da eine Manipulation des Datenflusses stattfindet, kann in diesem Fall von einem relationalen Präprozessor gesprochen werden [Aho88].

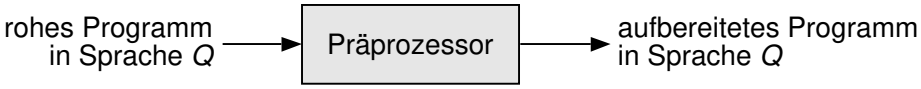


Abb. 6.2: Blockdiagramm eines relationalen Präprozessors

Um die Elementarbefehle möglichst einfach halten zu können, die Übersetzungszeit zu minimieren und den Befehlsvorrat der Zwischensprache gering zu halten, werden bei Bedarf folgende Funktionen durch den Präprozessor durchgeführt:

- **Schneidefunktion**
  Eine semantische Gliederung kann an besonderen Stellen aufgetrennt werden. Für das Auftrennen eignen sich alle Semune, die eine Vereinigung zweier Attribute oder Aktionen bewirken. Die Intention des Benutzers ist beim Aussprechen der Wortkette „die kugel und beide quader rot färben" sicherlich identisch mit den beiden aufeinander folgenden Wortketten „die kugel rot färben" und „beide quader rot färben". Wird eine Gliederung an einer „Sollbruchstelle" aufgetrennt, taucht dieses Semun, an dem aufgetrennt wurde, in den resultierenden Gliederungen nicht mehr auf.

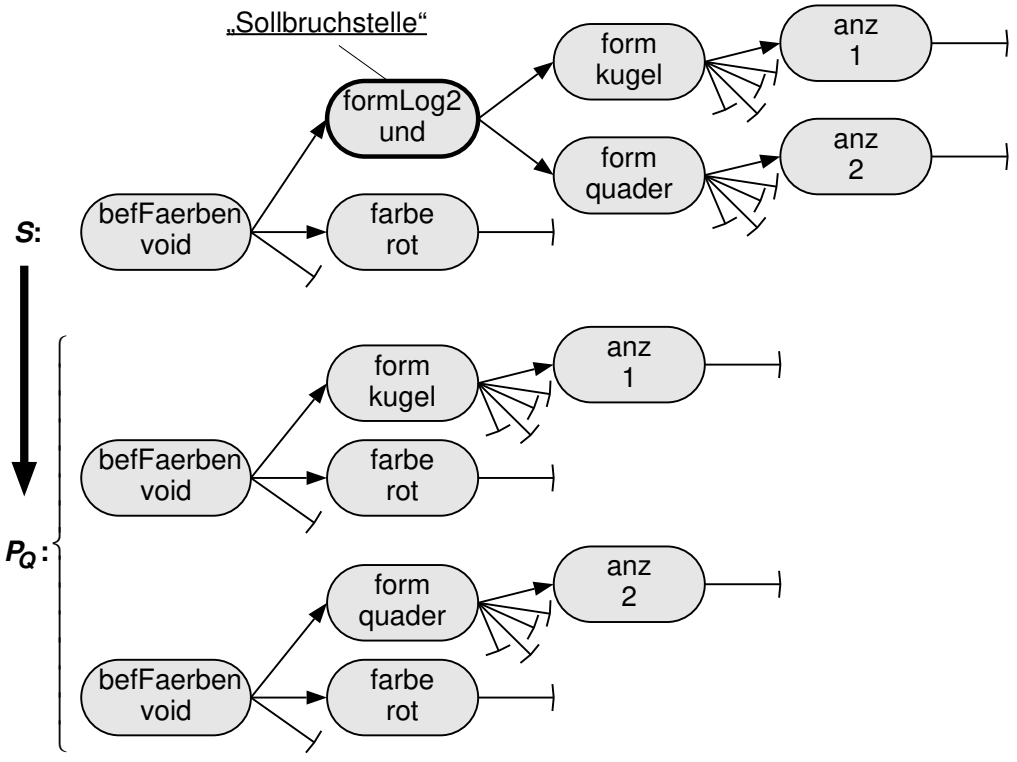


Abb. 6.3: Beispiel für die Anwendung der Schneidefunktion

Die beiden resultierenden Gliederungen bilden das zu compilierende Programm in der Quellsprache P_Q und enthalten die gesamte semantische Information der Eingangsgliederung. Das Auftrennen ist nicht immer von Vorteil. In einigen Situationen muß das Auftrennen unterbunden werden, um die semantische Information der Eingangsgliederung nicht zu verändern. Ein Auftrennen der semantischen Gliederung zur Wortkette „vertausche den quader und die kugel" in „vertausche den quader" und „vertausche die kugel" zerstört die Aussage der Eingangsgliederung. Daraus wird ersichtlich, daß durch die Behandlung von Ausnahmen das Eliminieren mancher Semune verhindert werden muß.

- **Einsetzfunktion**
  Zur konsistenten Übertragung in Elementarbefehle erwartet der nachfolgende Compiler nach Grafikeditierbefehlen stets eine Formangabe, d.h. Semune des Typs „form", „formAllg" bzw. „formBezug". Bei kurzen Anweisungen (z.B.: „rot färben") kann davon ausgegangen werden, daß das zuletzt bearbeitete Objekt, welches mit dem Semuntyp „formBezug" angesprochen werden kann, bearbeitet werden soll.

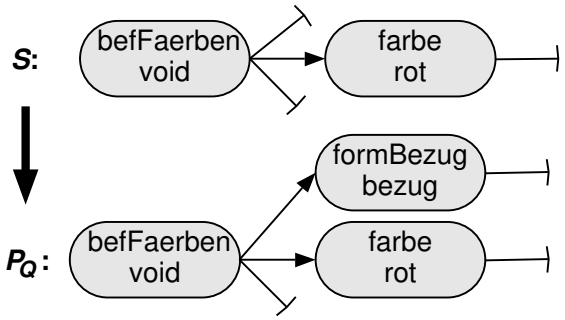


Abb. 6.4: Beispiel für die Anwendung der Einsetzfunktion

- **Optimierfunktion**
  Zur Einsparung von Rechenleistung, werden solche Semune eliminiert, die innerhalb der semantischen Gliederung keine Information beisteuern. Zum Beispiel findet bei einem Semun des Typs „befIrrelevant" und allen seinen Nachfolgern keinerlei Aktion am Rechner statt. Die redundanten Nachfolger-Semune werden eliminiert, wodurch eine kompakte Darstellung ermöglicht wird.

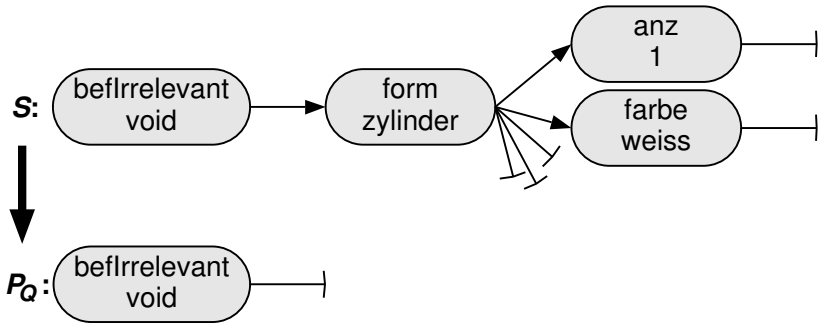


Abb. 6.5: Beispiel für die Anwendung der Optimierfunktion

6.3 Compiler

Eine oder mehrere vorverarbeitete semantische Gliederungen bilden das Programm in der Quellsprache für den in diesem Abschnitt beschriebenen Compiler. Diese Eingabe kann nicht direkt in Datenbankzugriffe umgesetzt werden, da der Rechner präzise und eindeutige Anweisungen benötigt. Dazu muß die innerhalb der semantischen Gliederung aufgrund der natürlichsprachlichen Kontextsensitivität vorhandene Mehrdeutigkeit erkannt und dementsprechend beseitigt werden.

Die formale Definition eines Compilers lautet [Bro95]:

> *Ein Übersetzer oder Compiler ist ein Programm, das Programme einer Quellsprache (source language) in ein semantisch äquivalentes Programm einer Zielsprache (target language) umwandelt.*

Umgangssprachlich ausgedrückt ist ein Compiler ein Modul, welches Programme in einer Quellsprache Q analysiert und anschließend in eine andere Zielsprache Z übersetzt.

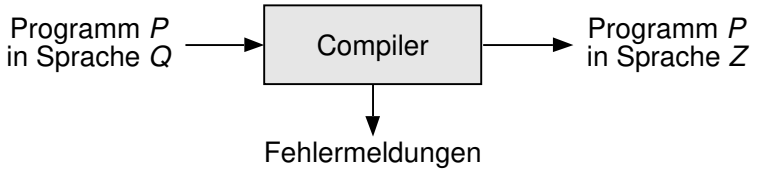


Abb. 6.6: Blockdiagramm eines Compilers

6.3.1 Kontextsensitivität

Die semantische Gliederung kann nicht (linguistisch betrachtet) kontextfrei sein, da sie als eine wortnahe Darstellung definiert ist und die menschliche Sprache hochgradig kontextsensitiv ist. Deshalb muß zuerst eine eventuell bestehende Kontextsensitivität erkannt und beseitigt werden, bevor die Übersetzung beginnen kann. Folgende Wortketten W_1 und W_2 sollen das Problem der Kontextsensitivität verdeutlichen:

- W_1: „verschiebe eine rote kugel nach rechts“
  Es soll eine existierende Kugel manipuliert werden. Diese Kugel muß unter allen Objekten, die sich augenblicklich auf dem Bildschirm befinden, herausgesucht werden.
- W_2: „zeichne eine rote kugel“
  Die Kugel ist ein Attribut für ein nicht existierendes Objekt, das erstellt werden soll. Nach einer Kugel darf hier keinesfalls gesucht werden.

In beiden Fällen wird das Wort „kugel“ wegen der sprachnahen Darstellung in der semantischen Gliederung mit demselben Semun „formAllg:kugel“ dargestellt. Nur aus der Umgebung (d.h. unter Berücksichtigung des Sinngehaltes der anderen Worte) läßt sich der Bedeutungsinhalt des Wortes „kugel“ vollständig erschließen. Daher werden das Zustandsmodell und das Zustandsübergangsmodell als notwendige Wissensbasen eingeführt, um das Umfeld eines Semuns zu beschreiben.

6.3.2 Analyse mit Zustandsmodell und Zustandsübergangsmodell

Die Darstellung jedes Semuns s_n wird innerhalb des Intentionsdecoders um den *Zustand* $z[s_n]$ erweitert. Die Menge aller möglichen Zustände ist im Zustandsmodell abgelegt und ist, wie auch das Typen- und Werteinventar, von der betrachteten Domäne abhängig. Für die ‚Grafikeditor'-Domäne erweisen sich folgende fünf Zustände als notwendig und zweckmäßig:

- Der Zustand „befehl" ist der übliche Startzustand der Gliederung.
- Der Zustand „neu" sammelt Attribute für die Erstellung neuer Objekte.
- Der Zustand „was" bestimmt immer ein Objekt oder eine Objektgruppe.
- Der Zustand „wieviel" hält quantitative Aussagen für die weitere Verarbeitung bereit.
- Der Zustand „wie" beinhaltet neue Eigenschaften.

Die Darstellung für ein Semun s_n, welches aus einem Typ $t[s_n]$, einem Wert $v[s_n]$ und einem Zustand $z[s_n]$ besteht, sei im folgenden:

$$t[s_n] : v[s_n] : z[s_n] \text{ oder } \begin{array}{c} t[s_n] \\ v[s_n] \\ \hline z[s_n] \end{array}.$$

Jedes Semun wird als ein Zustandsübergangsautomat aufgefaßt, der ausgehend von Typ $t[s_n]$ und Zustand $z[s_n]$ des Ausgangssemuns sowie der Nachfolgerposition x mit $1 \le x \le X$ in einen Nachfolgerzustand $z[q_x[s_n]]$ schaltet. Die Abfolge der Zustände für alle Typ-Zustands-Kombinationen enthält das Zustandsübergangsmodell.

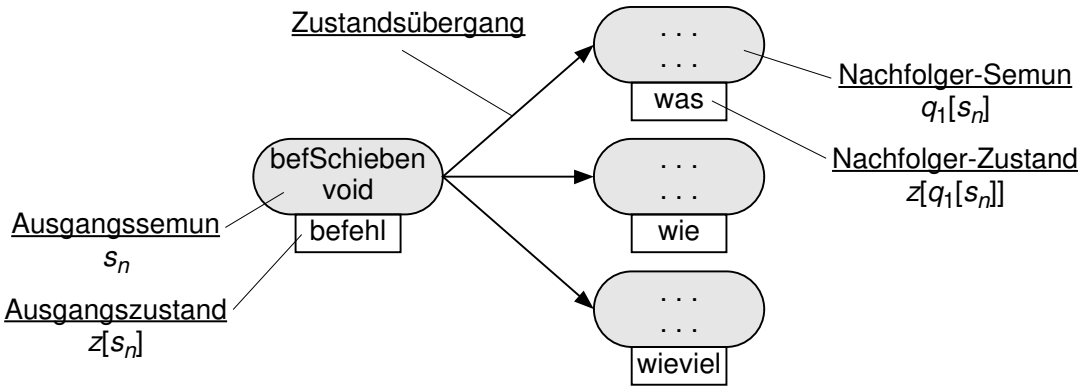


Abb. 6.7: Beispiel für einen Zustandsübergang

Ist das Semun mit dem Typ „befSchieben" im Zustand „befehl", dann befindet sich der erste Nachfolger im Zustand „was" (hier werden die Objekte erwartet, auf die agiert wird), der zweite im Zustand „wie" (wie werden die Objekte des was-Nachfolgers bearbeitet) und der dritte im Zustand „wieviel" (mit welcher Quantität werden die Objekte bearbeitet). Der Zustand „befehl" ist der übliche Startzustand, da die meisten Gliederungen mit einem Befehl eingeleitet werden. Bei der Abfolge der Zustände ist nur der Semuntyp entscheidend, da dieser die Anzahl der Nachfolger festlegt. Der Wert des Semuns liefert in diesem Zusammenhang keine weitere Information.

Während der nachfolgenden Zustandsanalyse wird die Bedeutung der Semune innerhalb der semantischen Gliederung S betrachtet. Um das kontextuelle Umfeld, in dem sich ein Semun s_n in der Gliederung S befindet, zu analysieren und zu modellieren, wird das Zustandsübergangsmodell herangezogen. Die gesamte Gliederung wird bis zu den leeren Nachfolgern vollständig durchlaufen und für jedes Semun s_n aus dem Zustandsübergangsmodell der aktuelle Zustand $z[s_n]$ ermittelt. Dieser aktuelle Semunzustand repräsentiert die augenblickliche Umgebung, in der sich das Semun befindet.

6.3.3 Zwischencode-Erzeugung

Das erworbene Wissen über das Umfeld jedes Semuns wird nun benutzt, um eine andere Darstellung der semantischen Gliederung zu erreichen. Nach der Zustandsanalyse ist jedem Semun s_n ein Typ $t[s_n]$, ein Wert $v[s_n]$ und ein Zustand $z[s_n]$ zugeordnet. Mit Hilfe dieser drei Parameter wird jedes Semun in eine Folge von Befehlen, im folgenden als Elementarbefehle bezeichnet, übersetzt. Die Befehle werden aus einer Wissensbasis, dem sogenannten *Befehlswissen*, bezogen.

Abb. 6.8: Ermittlung der Elementarbefehle mittels Befehlswissen

Nachdem jedes Semun in einer Elementarbefehlsfolge angegeben werden kann, wird die semantische Gliederung in einen Baum aus Elementarbefehlen umgesetzt. Aus dem semantischen Gliederungsbaum wird ein *Elementarbefehlsbaum*. Leere Nachfolger bleiben unberücksichtigt, da sie keine Information der semantischen Gliederung enthalten

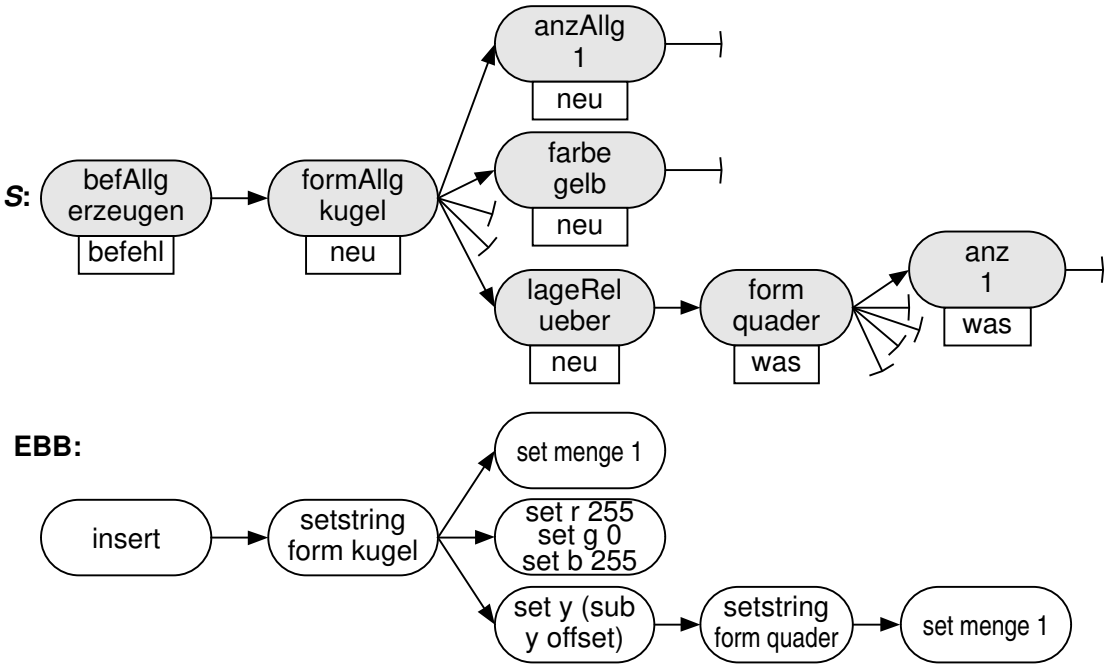


Abb. 6.9: Semantische Gliederung S mit korrespondierendem Elementarbefehlsbaum (EBB). Aus Übersichtgründen sind einige Elementarbefehle verkürzt dargestellt.

Die Elementarbefehle können anschließend auf dem Interpreter abgearbeitet werden. Die Ausführung dieser Befehle verändert das aktuelle Wissen des Interpreters, den sogenannten *Kontext*, der aus Variablen besteht. Um den Elementarbefehlsbaum in einer zeitlich definierten Abfolge abarbeiten zu können, muß dieser wie in Kap. 6.3.5 beschrieben linearisiert werden.

6.3.4 Kontext

Der Kontext wird vom Übersetzer und vom Interpreter gleichermaßen verwendet. Während der Übersetzer bei der Linearisierung den Kontext nur für allgemeine Operationen benutzt, wendet der Interpreter alle Elementarbefehle auf den Kontext an, indem er Werte von Variablen anfordert, diese modifiziert und wieder zurückschreibt oder Variablen durch Ergebnisse von Suchoperationen füllt. Der Kontext repräsentiert das augenblickliche Wissen des Systems, das fortlaufend bis zur endgültigen Abarbeitung der semantischen Gliederung verändert wird. Der Kontext birgt die Informationen für alle möglichen Operationen und besteht aus internen und externen Variablen:

- **Interne Variablen** sind „unsichtbare“ Variablen, die vom Benutzer nicht definiert werden können. Sie sind dem System von Anfang an bekannt, jedoch noch nicht gefüllt. Erst während der Abarbeitung des Baums werden sie vom Intentionsdecoder gefüllt oder zerstört. Interne Variablen sind zum Beispiel Ergebnisse von Suchanfragen an die Datenbank nach Objekten, die manipuliert werden sollen, oder spezifische Eigenschaften für neu zu erzeugende Objekte.
- **Externe Variablen** kann der Benutzer (oder besser ein kundiger Systemverwalter) frei definieren, mit Werten belegen und in seine Berechnungen miteinbeziehen. Beispiele wären die Bildschirmbreite, die den Bereich für Suchanfragen mitbestimmt oder Defaultwerte, die eingesetzt werden, falls der Bediener wichtige Daten nicht näher benennt. Bei der Äußerung „schiebe den quader“ wird weder der Betrag der Translation noch deren Richtung bestimmt. Hier müssen durch externe Variablen definierte Defaultwerte eingesetzt werden.

6.3.5 Linearisierung des Elementarbefehlsbaums mit ‚top-up‘-Ansatz

Um den Elementarbefehlsbaum interpretergerecht aufzubereiten, muß er in eine lineare Aneinanderreihung von Elementarbefehlen umgeformt werden, die auf dem nachfolgenden Interpreter Befehl für Befehl abgearbeitet wird. Die Nachfolgezweige eines Befehlssemuns werden als *Unterbäume* bezeichnet. Der Name des Unterbaums ergibt sich aus dem Startzustand dieses Unterbaums. Hier muß von den Standardansätzen des Compilerbaus (Abarbeitung in ‚top-down‘[23]- bzw. in ‚bottom-up‘-Richtung [Gol90]) abgewichen werden. Denn um den aktuellen Zustand eines jeden Semuns zu bestimmen, muß der

23) Das ‚top-down‘-Parsing beginnt mit der Wurzel des Parse-Baumes und arbeitet sich ‚herunter‘ zu den Blättern. Beginnend mit der Hypothese, daß der Quelltext ein gültiges Programm ist, wird versucht, diese Annahme zu bestätigen, indem die Untereinheiten des Quelltextes daraufhin untersucht werden, ob sie Elemente syntaktischer Kategorien der ersten Programmebene sind.

Baum ‚top-down' mit Hilfe des Zustandsübergangsmodells abgearbeitet werden. Die Elementarbefehle können aber nur ‚bottom-up'[24] generiert werden, da sie in höheren Ebenen des Baumes nur ausgeführt werden können, wenn die Elementarbefehle der Nachfolgersemune ausgeführt worden sind, da diese nähere Informationen liefern. Deshalb werden beide Verfahren zu einem ‚top-down-bottom-up'-Ansatz, kurz ‚top-up'-Ansatz, kombiniert.

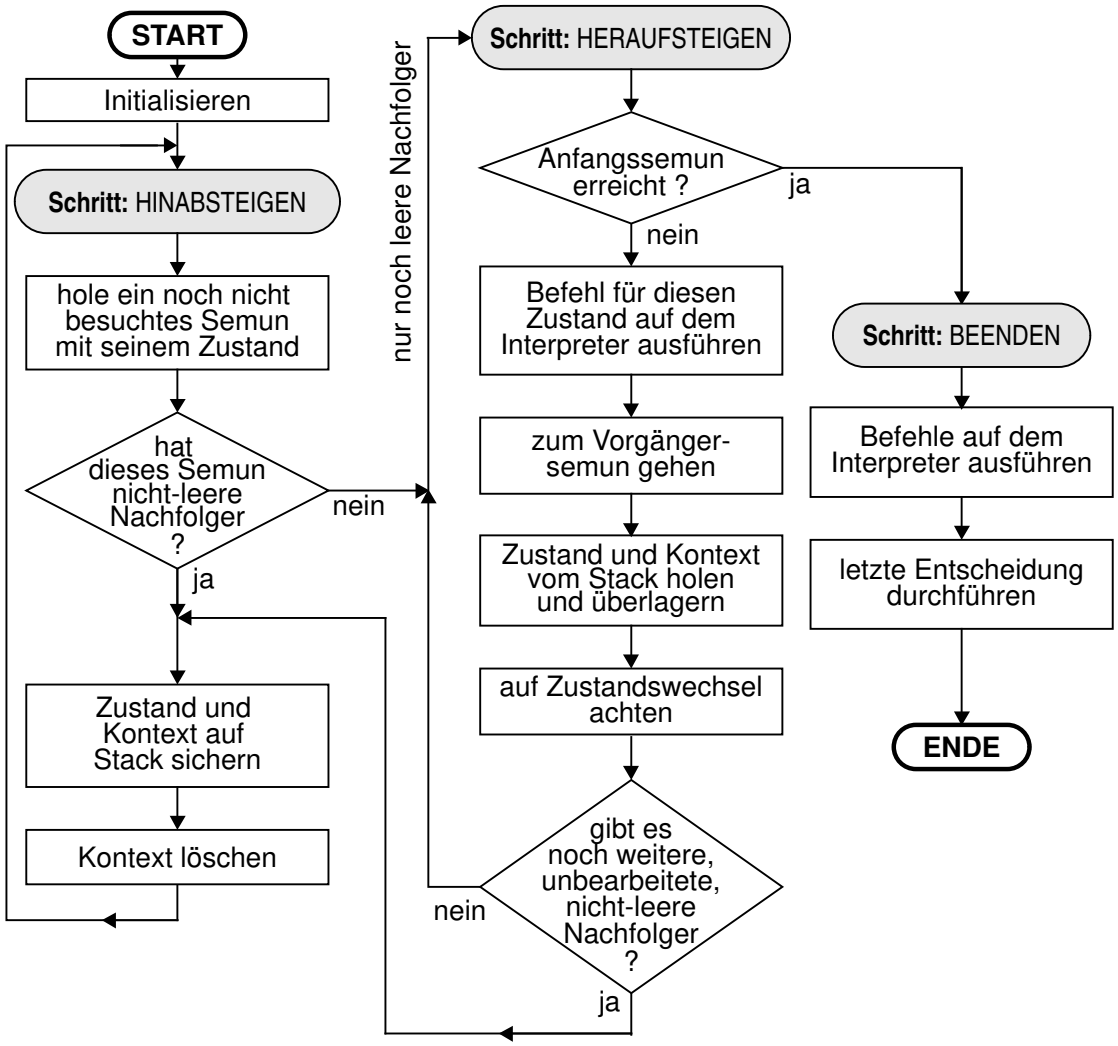


Abb. 6.10: Flußplan des Unterbaum-Linearisierungsalgorithmus'

In Abb. 6.10 ist der Flußplan für die Abarbeitung eines Unterbaums abgebildet mit den wesentlichen Schritten „HINABSTEIGEN" für die ‚top-down'-Abarbeitung des Baumes, „HERAUFSTEIGEN" für die ‚bottom-up'-Generierung der Elementarbefehle und „BEENDEN" für die ultimative Weitergabe an den Interpreter. Eine ausführliche Beschreibung dieses Vorgehens wird in [Ebe95] aufgezeigt. Der Ansatz soll im folgenden beispielhaft mit der

24) Das ‚bottom-up'-Parsing beginnt mit den Blättern des Parse-Baumes und arbeitet sich hoch zur Wurzel. Dabei werden aus benachbarten Tokens des Quellprogramms einfache syntaktische Kategorien geformt. Diese Elemente werden dann kombiniert, um Elemente komplexerer Kategorien zu bilden.

in Abb. 6.11 dargestellten Äußerung verdeutlicht werden, bei welcher der neu-Ast $S(s_2)$ linearisiert wird:

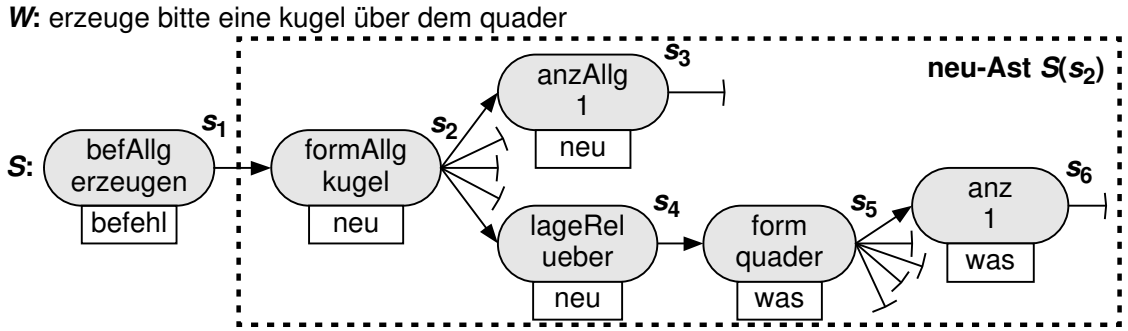
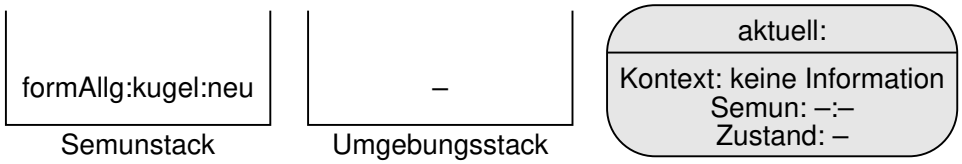


Abb. 6.11: Wortkette W und semantische Gliederung S einer Äußerung mit Ast $S(s_2)$. In der Hierarchie wird die Lage der weiter links liegenden Semune als „höher“, die der weiter rechts liegenden Semune als „tiefer“ bezeichnet.

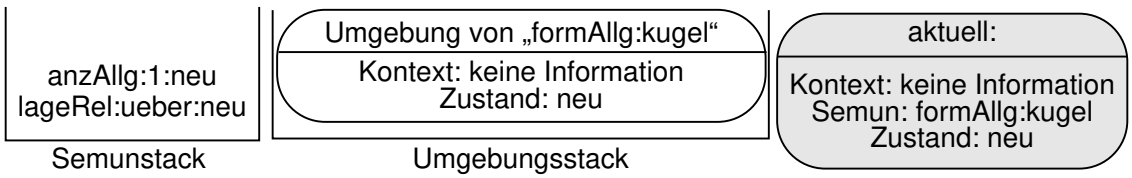
In den folgenden Ausführungen ist im grau unterlegten Kasten mit der Aufschrift „aktuell“ das gerade zu bearbeitende Semun mit seinem Zustand und dem augenblicklichen Wissen des Kontexts aufgelistet.

1. Schritt: Es werden verschiedene Stacks initialisiert und eingerichtet:

- Ein Semunstack, der mittels eines modifizierten Tiefensuche-Algorithmus‘ [Lan95] nacheinander bisher unbearbeitete Semune liefert. Wird ein Semun von diesem Stack geholt, dann werden dort alle seine nicht-leeren Nachfolger mit jeweiligem Zustand von unten nach oben abgelegt.
- Ein Umgebungsstack für das Umgebungswissen eines Semuns, bestehend aus Kontext und Semunzustand.



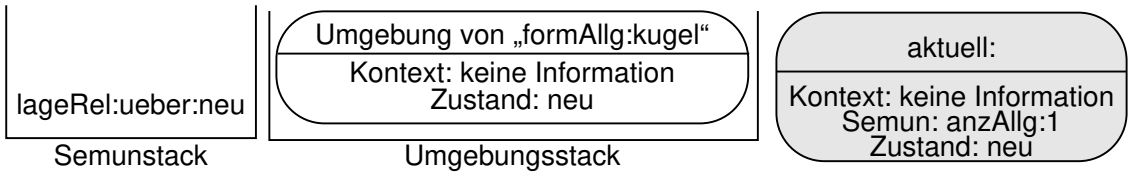
2. Schritt: Das neue, aktuelle Semun wird mit seinem Zustand vom Semunstack geholt. Automatisch werden alle Nachfolgersemune mit ihren Zuständen auf diesen Stack gelegt. Da das Semun „formAllg:kugel“ nicht-leere Nachfolger besitzt, werden der aktuelle Kontext und der aktuelle Zustand auf den Umgebungsstack gelegt. Danach wird der aktuelle Kontext zerstört, um kein Wissen in tiefergelegene Zweige des Unterbaums zu transportieren.



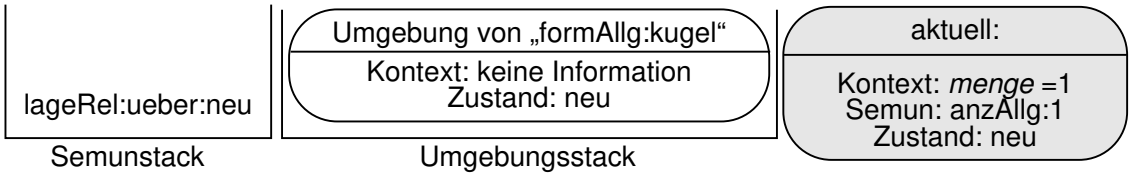
Beim Verzweigen zu „weiter rechts“ gelegenen Semunen darf kein Kontextwissen von „links“ nach „rechts“ transportiert werden. Information, die von „links“ nach „rechts“ ge-

langt, würde die rechten Ebenen vor allem bei Suchoperationen nachteilig beeinflussen. Daher muß der Kontext nach Ablage auf einem Stack zerstört werden. (Zum Beispiel wäre bei „schiebe die rote kugel über den quader" nach der Abarbeitung der Semune zu „rote kugel" im Kontext „rot" enthalten. Wird diese Information nicht gelöscht, würde fälschlicherweise ein roter Quader gesucht werden.)

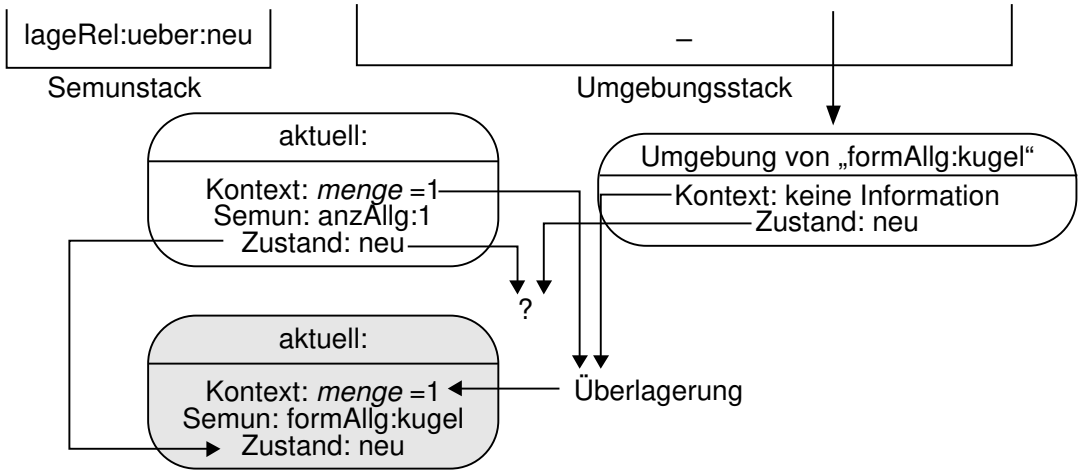
3. Schritt: Das Semun, das vom Semunstack geholt wird, besitzt keine weiteren Nachfolger, die auf den Stack gelegt werden müßten. Deshalb wird mit dem aktuellen Semun nach „HERAUFSTEIGEN" (siehe Flußplan, Abb. 6.10) verzweigt.



4. Schritt: Da es sich nicht um das Anfangssemun des Unterbaums handelt, wird der Elementarbefehl für das aktuelle Semun im Zustand „neu" ausgeführt. Das heißt, die Variable *menge* wird durch Abarbeitung des entsprechenden Befehls mit dem Wert 1 belegt.



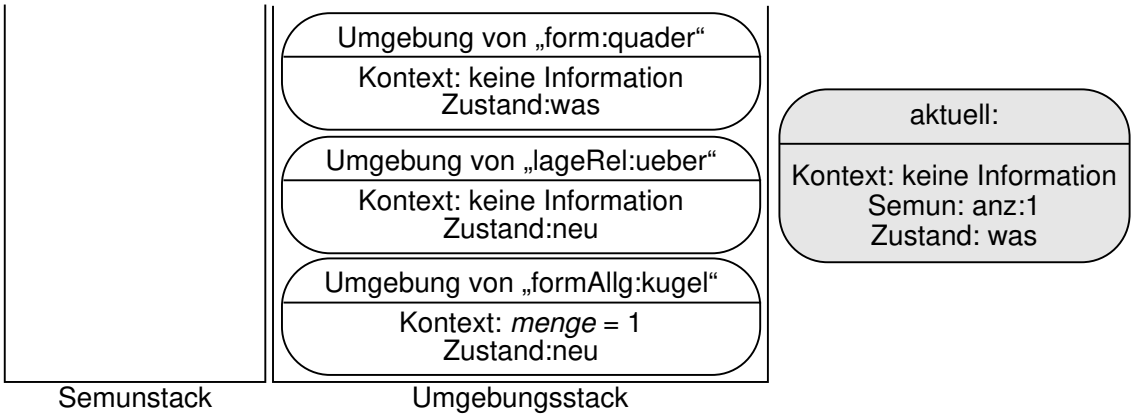
5. Schritt: Beim Zurückgehen zum Vorgängersemun wird vom Umgebungsstack der vorher gesicherte Kontext und der Semunzustand geholt. Jetzt werden der aktuelle Semunzustand und der Semunzustand, der vom Umgebungsstack geholt wurde, verglichen. Da beide Semunzustände gleich sind, findet keine Sonderaktion statt. Der aktuelle Kontext und der des Umgebungsstacks werden überlagert, das heißt das Wissen aus beiden Kontexten wird zusammengefaßt.



6. Schritt: Das aktuelle Semun „formAllg:kugel" besitzt noch einen unbearbeiteten, nichtleeren Nachfolger. Deshalb wird nach dem Sichern der Umgebung der Kontext des aktuellen Semuns zerstört, um kein Wissen in tiefere Zweige des Unterbaums zu transportie-

ren. Nun wird durch wiederholtes Aufrufen des Schrittes „HINABSTEIGEN“ bis zum Semun „anz:1“ gegangen, und die entsprechenden Daten werden auf die Stacks gelegt.

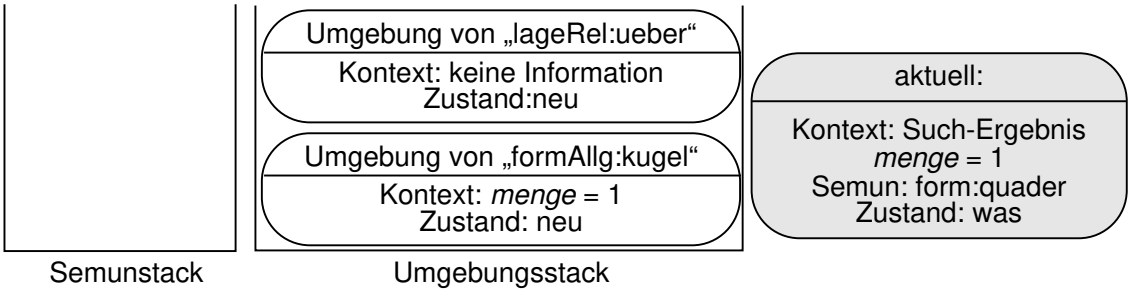
7. Schritt: Vom Semunstack wird das Semun „anz:1“ mit dem Semunzustand „was“ geholt. Dieses Semun hat keine unbearbeiteten, nicht-leeren Nachfolger. Deshalb verzweigt das Programm gemäß Flußplan in den Schritt „HERAUFSTEIGEN“.



8. Schritt: Da es sich bei dem aktuellen Semun nicht um das Startsemun handelt, kann der zugehörige Befehl für den Semunzustand „was“ abgearbeitet werden. Die Variable *menge* wird durch Abarbeitung des entsprechenden Befehls auf den Wert 1 gesetzt.

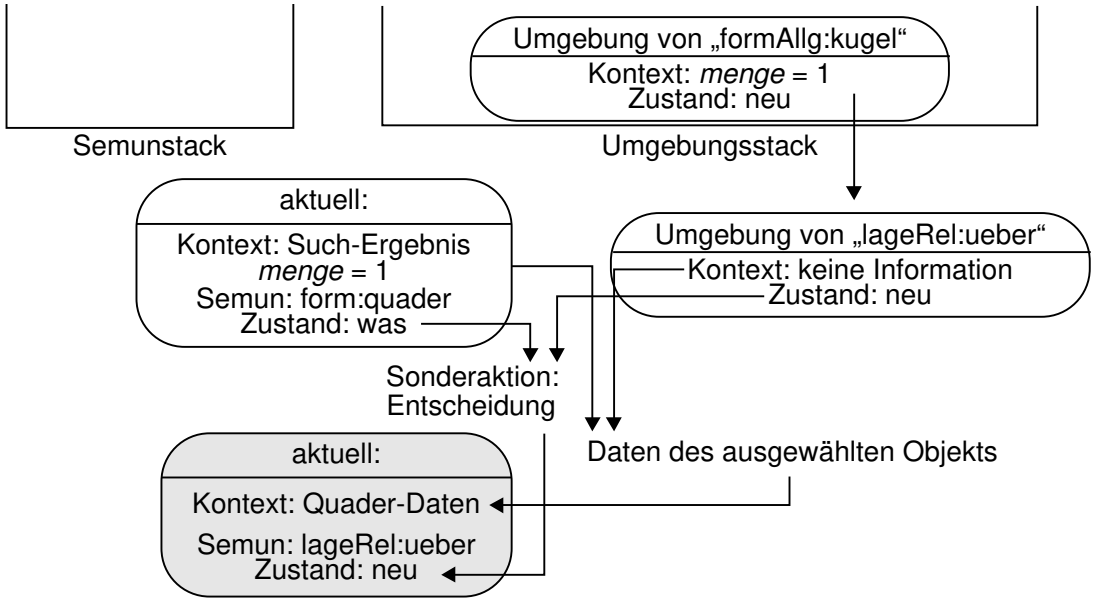
9. Schritt: Nun wird im Unterbaum wiederum zum Vorgängersemun „form:quader“ gegangen. Vom Umgebungsstack wird der vorher gesicherte Kontext und der Semunzustand geholt. Dann werden der aktuelle Semunzustand und der Semunzustand, der vom Umgebungsstack geholt wurde, verglichen. Da beide Semunzustände gleich sind, findet keine Sonderaktion statt. Der aktuelle Kontext und der des Umgebungsstacks werden überlagert, das heißt zusammengefaßt.

10. Schritt: Das aktuelle Semun besitzt keine unbearbeiteten, nicht-leeren Nachfolger, und es handelt sich nicht um das Anfangssemun des Unterbaums. Deshalb wird es mit dem aktuellen Semunzustand auf dem Interpreter ausgeführt. Es findet eine Suche nach einem Quader statt.



11. Schritt: Es wird zum Vorgängersemun gesprungen. Das Semun „lageRel:ueber“ wird zum aktuellen Semun. Dabei wird vom Umgebungsstack der vorher gesicherte Kontext und der Semunzustand geholt. Anschließend werden der aktuelle Semunzustand und der Semunzustand, der vom Umgebungsstack geholt wurde, verglichen. Da beide Semunzustände verschieden sind, muß eine Sonderaktion stattfinden. Der Zustand „was“ hat eine

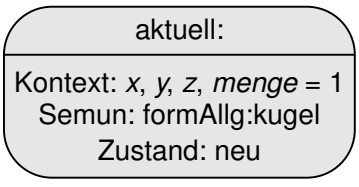
Objektgruppe oder ein Objekt ermittelt, im Zustand „neu“ werden jedoch Ortskoordinaten erwartet. Deshalb muß sich das System bei diesem Zustandsübergang für die gewünschte Anzahl an Quadern entscheiden. Entscheidungen dürfen nur bei Zustandswechseln durchgeführt werden. Eine Suchanfrage an die Datenbank darf sich nie entscheiden. (Bei „... die kugel über dem quader ...“ können mehrere Quader gefunden werden, obwohl nur einer gesucht wurde; es existiert aber möglicherweise nur eine Kugel direkt über einem Quader.) Nun wird der gewünschte Quader aus dem Ergebnis der Datenbankanfrage herausgesucht und dessen Daten in den Kontext übertragen.



12. Schritt: Da es keine nicht-leeren, unbearbeiteten Nachfolger mehr gibt, wird in diesem Ausführungszustand weitergearbeitet. Es handelt sich nicht um das Startsemun, deshalb werden die folgenden Elementarbefehle auf dem Interpreter ausgeführt:

- Nimm den Hochwert abzüglich einem Offset als y-Komponente.
- Nimm den Rechtswert als x-Komponente für die Erzeugung.
- Nimm den Tiefenwert als z-Komponente für die Erzeugung.
- Alle anderen Werte werden für ungültig erklärt und können nicht mehr im weiter links befindlichen Teil der Gliederung verwendet werden. Das in weiter rechts befindlichen Teilen des Baumes erworbene Wissen wird somit für linksgelegene Baumteile gefiltert (außer den Ortskoordinaten sind ab diesem Semun im linken Teil des Baumes keine Informationen mehr bekannt).

13. Schritt: „formAllg:kugel“ wird zum aktuellen Semun, indem zum Vorgängersemun aufgestiegen wird. Dabei werden vom Umgebungsstack wieder der vorher gesicherte Kontext und der Semunzustand geholt. Der aktuelle Semunzustand und der Semunzustand, der vom Umgebungsstack geholt wurde, werden verglichen. Da beide Semunzustände gleich sind, findet keine Sonderaktion statt. Beide Kontexte werden überlagert. Da dieses Semun keine unbearbeiteten Nachfolger mehr hat und zudem das Anfangssemun des Unterbaums ist, wird zum Schritt „BEENDEN“ verzweigt.



14. Schritt: Der Befehl des aktuellen Semuns „formAllg:kugel“ wird ausgeführt: Der Variablen *form* wird die jeweilige Körperform zugeordnet. Nun ist die semantische Information des Unterbaums ausgewertet. Die Variablen *form*, *x*, *y*, *z* und *menge* werden dem Befehlssemun übergeben. Die restlichen noch benötigten Daten werden durch den Interpreter mit Defaultwerten aufgefüllt. Dort wird der mit dem Befehlssemun verbundene Elementarbefehl ausgeführt, der ein Einsetzen eines neuen Grafikobjekts in die Datenbank bewirkt. Die Abarbeitung der semantischen Gliederung ist somit beendet. Die Datenbank oder Grafikdatenbasis wird anschließend herangezogen, um den Bildschirmaufbau zu aktualisieren. Sollten während der Abarbeitung der Gliederung Sinnhaftigkeits- oder Eindeutigkeitsfehler auftreten, wird eine Fehlermeldung in die Grafikdatenbasis eingetragen, die über ein Textfenster optisch oder mit synthetisierter Stimme akustisch ausgegeben wird.

6.4 Der Interpreter

Die formale Definition für einen Interpreter lautet [Bro95]:

> *Ein Interpreter ist ein Programm, das Programme ausführen kann, die in seiner Interpretersprache formuliert sind.*

Aho et al. merken dazu folgendes an [Aho88]:

> *Anstatt ein Zielprogramm als Übersetzung zu erzeugen, führt ein Interpreter die im Quellprogramm enthaltenen Operationen direkt aus.*

Ein Interpreter simuliert die Ausführung eines Programms, das in seiner Interpretersprache formuliert ist. Der Interpreter vermittelt die Illusion, eine Maschine zur Verfügung zu haben, welche die neue Sprache „versteht“. Die Ausführung eines derartigen Interpreterprogramms liefert, sieht man von der Geschwindigkeit ab, exakt die gleichen Resultate, die dessen Ausführung auf einem Rechner liefern würde, der tatsächlich diese Sprache als Maschinensprache besitzt. Aus diesem Grund spricht man in diesem Zusammenhang auch von einer *virtuellen Maschine* [Bli92].

Die Semune der semantischen Gliederung wurden in Elementarbefehle umgesetzt, die auf dem Interpreter ausgeführt werden. Die Umsetzung dieser Kommandos in eine entsprechende Grafik-Datenbasis soll dem Wunsch des Benutzers, d.h. dessen Intention, möglichst genau entsprechen. Um der Aufgabenstellung nach einer möglichst flexiblen Lösung nachzukommen, muß eine Interpretersprache mit folgenden Anforderungen geschaffen werden:

- Die Sprache muß einfach anzuwenden sein.
- Sie muß die Flexibilität besitzen, auch zukünftigen Anforderungen zu genügen.
- Sie muß eine schnelle und einfache Abarbeitung gewährleisten.
- Der Systembetreuer muß in der Lage sein, Entscheidungen in dieser Sprache zu formulieren, um zum Beispiel die räumliche Lage der Objekte untereinander zu berücksichtigen.

- Die Sprache muß Berechnungen durchführen zu können.
- Die Sprache muß eine Datenbank bearbeiten können.
- Die Sprache muß Möglichkeiten bereitstellen, den Datenfluß zu regeln.
- Die Sprache muß Schnittstellen zur Ausgabe von Meldungen aus dem Compiler-Interpreter-System bereitstellen.

Diese Sprache wird aus der Analyse der Grafikdatenbasis und eines bereitgestellten Äußerungsschatzes abgeleitet. Dieser enthält semantische Gliederungen, die aus 1843 umgangssprachlichen Äußerungen geformt wurden. Um arithmetische Operationen in Präfixnotation abzuarbeiten, wurde eigens dafür ein Formelparser erstellt. Für den gezielten Zugriff auf die Grafik-Datenbank sorgen weitere spezielle Befehle. Die genaue Syntax der entworfenen Interpretersprache wird detailliert in [Ebe95] hergeleitet und beschrieben, so daß an dieser Stelle auf eine erneute Darstellung verzichtet wird.

Entscheidungen werden benötigt, um den linearen Ablauf der Abarbeitung gegebenenfalls zu unterbrechen oder um von Bedingungen abhängige Maßnahmen durchzuführen. Dabei können abhängig von einer Bedingung verschiedene Interpreterbefehle ausgeführt werden. Als Beispiel dient die in Abb. 6.12 gegebene Konstellation, bei der die Eingabe-Wortkette „schiebe den quader neben die kugel“ lautet. Für die Ausführung dieser Äußerung muß zunächst die betreffende Kugel gefunden werden, neben welcher der Quader plaziert werden soll. Nun muß unterschieden werden, ob sich der Quader vor Befehlsausführung rechts oder links neben der Kugel befindet. Abhängig davon wird nun der Quader entweder rechts neben oder links neben der Kugel placiert.

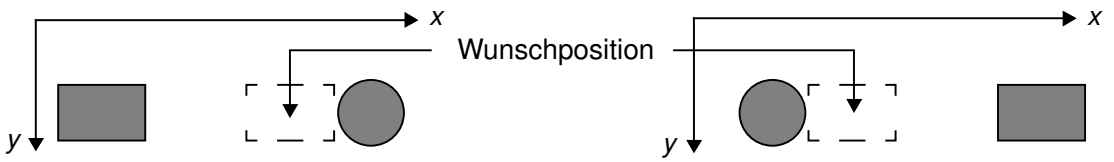


Abb. 6.12: Bildschirmbeispiele für eine abhängige Entscheidung beim Wort „neben“

6.5 Weitere Verarbeitung

Der Pixelgenerator setzt die vom Interpreter aktualisierte Grafikdatenbasis in eine Pixelsequenz um. Im Rahmen der vorliegenden Arbeit wurde das Softwarepaket *SIPP* [25)] zur Erstellung dreidimensionaler Objekte verwendet [Yng92]. Jede einzelne Zeile der Grafikdatenbasis wird sequentiell eingelesen, und die nötigen Befehle zur Darstellung aller existierender Objektes werden ausgeführt. Mit *SIPP* werden die generierten Pixelsequenzen in einer Datei im ppm-Format gespeichert.

25) SIPP ist ein universelles ‚public domain‘-Bibliothekenprogramm, das die Darstellung dreidimensionaler Objekte mit sowohl vordefinierten als auch frei definierbaren Funktionen in der Programmiersprache C erleichtert. Die einstellbaren Möglichkeiten beinhalten eine größere Anzahl von Objektformen, Oberflächen-Texturen, Beleuchtungsarten und Schattierungen, von denen jedoch nur ein kleiner Teil aus Gründen der sprachlichen Einschränkungen im implementierten Grafikeditor genutzt wird.

Die Visualisierung dieser ppm[26)]-Datei kann im nachfolgenden Programmschritt über ein Standard-Pixelausgabeprogramm[27)] erfolgen. Alternativ dazu wurde innerhalb der Arbeit von [Sta97b] eine interaktive Benutzeroberfläche mit der Skriptsprache *Tcl/Tk* [Wel95] implementiert, die neben einer komfortablen Menüsteuerung eine Visualisierung von ppm-Dateien ermöglicht.

6.6 Ergebnis und Diskussion

Das gesamte Compiler-Interpreter-System wurde mit 1843 semantischen Gliederungen, die mit einer „Wizard of Oz"-Simulation gesammelt wurden [Mül95c], getestet. Es ergab sich eine subjektiv korrekte Ausführungsrate von 97,3%.

Gemäß Pieraccini et al. ist es um so schwieriger, eine semantische Repräsentation von Sprache auf einem Rechner ausführbar zu machen, je sprachnäher diese Repräsentationsform ist [Pie93]. Die semantische Gliederung erhebt jedoch den Anspruch auf eine wortnahe Darstellung [Mül95a]. Aus diesen Gründen ist es wohl nicht möglich, eine 100% korrekte Ausführungsrate zu erhalten.

Die Einheitlichkeit aller Interpreterbefehle hinsichtlich der übergebenen Daten und die einfache Integration neuer Funktionen erlauben dem Systembetreuer, schnell und effektiv neue Funktionen einzufügen. Durch die Mächtigkeit der Interpreterbefehle und dem um externe Variablen erweiterbaren Kontext kann das System sowohl auf Spezialfälle des Grafikeditors angepaßt als auch auf andere Domänen portiert werden. Die rechnerinterne, numerische Darstellung aller Daten, die bei jedem Programmlauf neu berechnet wird, erlaubt es dem Systembetreuer, schnell und einfach neue Semune oder weitere Zustände in das Gesamtsystem zu integrieren.

26) Das Kürzel *ppm* bedeutet *portable pix(el) map* und bezeichnet ein standardisiertes Datenformat für zweidimensionale, orts- und amplitudendiskrete Bilder.

27) Dazu eignen sich beispielsweise die Programme *xv* oder *ximagetool*.

Kapitel 7

Verbesserung der Akzeptanz durch benutzeradäquaten Dialog

Gerade bei einer neuen Technologie wie der Sprachverarbeitung ist ein ungeübter Benutzer besonders kritisch – erwartet er doch ein neues und damit seiner Meinung nach verbessertes System. Deshalb ist neben der Funktionalität und der Robustheit in erster Linie auf die Bedienbarkeit und auf eine benutzerseitige Akzeptanz zu achten. In diesem Kapitel soll neben einer Erklärung des Akzeptanz-Begriffes und möglichen Fehlerursachen auch auf den Aufbau von sprachlichen Mensch-Maschine-Dialogen eingegangen werden, insbesondere im Hinblick darauf, mit einem geeigneten Dialog Decodierungsfehler vorab zu vermeiden oder nachträglich zu korrigieren.

7.1 Was ist ‚Akzeptanz'?

Allgemein bedeutet der Begriff *Akzeptanz* die „Bereitschaft, etwas zu akzeptieren", wobei *akzeptieren* im positiven Sinn von „annehmen, billigen, hinnehmen" umschrieben werden kann [Dro90].

Wenn es jedoch im hier betrachteten Zusammenhang um die Akzeptanz eines Benutzers gegenüber einem technischen System (sowohl Hard- als auch Software) geht, muß diese allgemeine Definition näher spezifiziert werden. So muß *Akzeptanz* definiert werden

- bezüglich der inneren Einstellung eines Benutzers gegenüber einer Technologie oder eines technischen Systems und
- bezüglich des Verhaltens eines Benutzers im Umgang mit dieser Technologie oder diesem technischen System.

Diesen prinzipiellen Kriterien entsprechend kann also zwischen einer *Einstellungsakzeptanz* uns einer *Verhaltensakzeptanz* unterschieden werden [Mül86]. Durch folgende prägnante Definition werden diese zwei wesentlichen Merkmale ebenfalls verdeutlicht:

> *Akzeptanz ist die positive Einstellung zur Technik und aufgabenbezogene Nutzung der zur Verfügung gestellten Funktionen* [Rei81].

Die Akzeptanz einer Technologie kann gefördert werden durch effizientere Technik, leichtere Bedienung, angepaßtere Funktionalität, höhere Verarbeitungsgeschwindigkeit, moderneres Outfit, günstigeren Preis, größere Fehlerrobustheit, geringere Störanfälligkeit und durch eine technologisch-innovativ bedingte Faszination[28]. Der letztere Punkt ist bei einer natürlichsprachlichen Informationseingabe sicherlich gegeben, ob jedoch die anderen Punkte – insbesondere Fehlerrobustheit – erfüllt sind, bleibt fraglich.

Im Unterschied zu anderen Arbeiten soll in diesem Kapitel die Akzeptanz nur als qualitatives Ziel, welches grundsätzlich angestrebt werden sollte, nicht jedoch als quantitativ zu ermittelnde Meßgröße verstanden werden.

7.2 Decodierungsfehler

Da die Spracherkennungs- bzw. Sprachverstehenstechnologie derzeit noch fehlerbehaftet arbeitet, d.h. solange sich die erzielbaren Erkennungsraten (zum Teil weit) unter 100% befinden, ist der Umgang mit unvermeidbar auftretenden Fehlern und Problemen ein entscheidender Faktor für die Akzeptanz eines sprachverarbeitenden Systems. Eine gezielte Fehlervermeidung und Fehlerkorrektur ist bei der Gestaltung eines sprachlichen Mensch-Maschine-Dialoges ein wichtiger Aspekt [Har93].

In der Spracherkennung (d.h. in der reinen Wortkettendecodierung) können folgende Arten von Fehlern unterschieden werden:

- **Vertauschung (‚substitution'):** In diesem Fall wird ein anderes, falsches Wort anstelle des wirklich gesprochenen Wortes detektiert.
- **Auslassung (‚deletion'):** Ein gesprochenes Wort wurde verworfen und taucht somit in der textuellen Ausgabe nicht mehr auf.
- **Einfügung (‚insertion'):** Ein nicht gesprochenes Wort wurde eingefügt.[29]

Diese Fehlerarten lassen sich auf den in dieser Arbeit beschriebenen Prozeß der semantischen Decodierung, die eine aus mehreren Semunen bestehende semantische Gliederung ausgibt, übertragen. Analog zur aus mehreren Worten bestehenden Wortkette können auch innerhalb der semantischen Gliederung einzelne Semune vertauscht, ausgelassen oder eingefügt worden sein, wodurch die erkannte semantische Gliederung mehr oder minder verfälscht ist. Die denkbaren Ursachen der auftretenden Decodierungsfehler kön-

28) Gerade Personen mit ausgeprägter Einstellungsakzeptanz gegenüber technologischen Innovationen „verzeihen" unter Umständen einem neuen System einige Nachteile nur aufgrund der Begeisterung über die neue bzw. ungewohnte Technologie.

29) In diesem Zusammenhang seien *SUB* die auf die Gesamtzahl aller Worte bezogene Prozentrate der Vertauschungen, *DEL* diejenige der Auslassungen und *INS* diejenige der Einfügungen. Daraus lassen sich folgende Größen bestimmen:
Fehlerrate (‚error rate') = *SUB* + *DEL* + *INS*
Akkuratheit (‚accuracy') = 100% – *SUB* – *DEL* – *INS*
Erkennungsrate (‚recognition rate') = 100% – *SUB* – *DEL*

nen, wie in den umseitigen Punkten aufgelistet, von verschiedenartigster Gestalt und in allen Verarbeitungsebenen angesiedelt sein:

- Bei einem sprecherunabhängigen System stimmt die Aussprache (z.B. Undeutlichkeit, Dialekt, ausländischer Akzent) mit den verwendeten akustischen und phonetischen Modellen nicht überein.
- Es treten spontansprachliche Effekte auf, wie Wortwiederholung, Wortauslassung, Satzabbruch, Räuspern oder Stottern. Diese können von der Grammatik nicht modelliert werden.
- Der Sprecher verwendet ein dem Systemvokabular unbekanntes Wort (d.h. ‚Out-of-Vocabulary' – OOV).
- Bei einem grammatikalisch restriktiven System[30)] erfüllt eine gesprochene Wortkette nicht die Anforderungen der Grammatik; sie kann syntaktisch nicht decodiert werden.
- Zur semantischen Decodierung verwendet der Sprecher eine dem System unbekannte semantische Konstellation.
- Umgebungsgeräusche werden als gesprochene Worte interpretiert.
- Durch unzureichendes Trainingsmaterial sind die in stochastischen Wissensbasen enthaltenen Wahrscheinlichkeiten unzureichend gewichtet oder die in regelbasierten Wissensbasen enthaltenen Inferenzen unvollständig. Dies führt beim Decodierungsprozeß zu Fehlentscheidungen.

Folgende Abbildung faßt die möglichen Fehlerquellen der semantischen Decodierung anschaulich zusammen und zeigt die jeweiligen Konsequenzen auf.

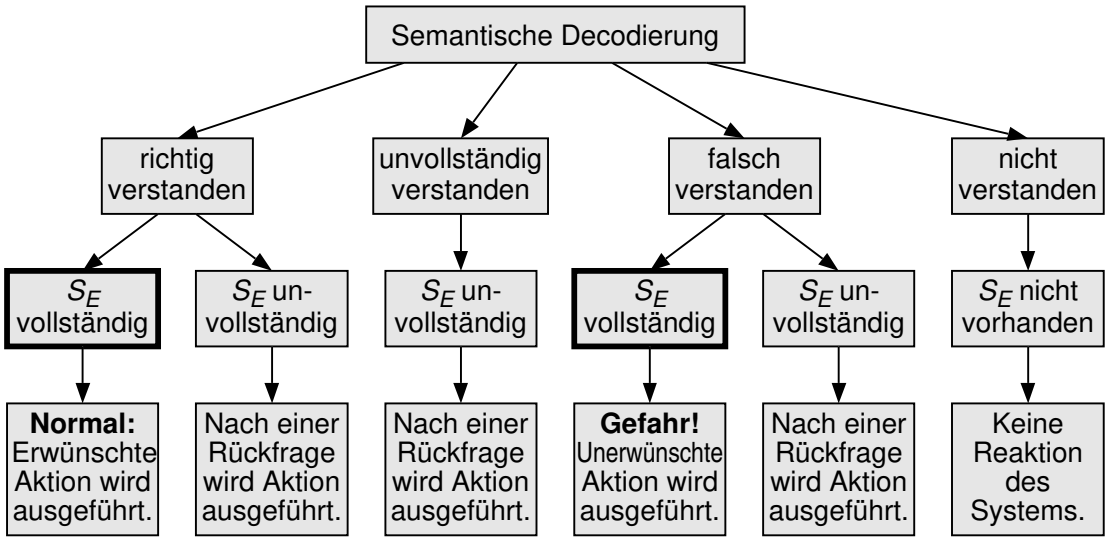


Abb. 7.1: Fehlerquellen der semantischen Decodierung [Hav96]

30) Üblicherweise sind Systeme zur Erkennung fließender Sprache mit einer Grammatik, d.h. einem Sprachmodell (‚language model'), zur sinnvollen Restriktion der Wortabfolge versehen. Dafür werden derzeit bevorzugt *n*-gramm-Modelle, welche die Abfolge von *n* aufeinander folgenden Worten stochastisch beschreiben, verwendet. In dieser Arbeit ist die Kombination aus semantischem und syntaktischem Modell eine Grammatik [Sta95], die sich auf die erlaubte Syntax beschränkend auswirkt.

Das Auftreten von Erkennungs- oder Interpretationsfehlern kann je nach Applikation mehr oder weniger gravierende Folgen mit sich führen. Bei der Applikation „Hörende Schreibmaschine“, also der reinen Konvertierung von Sprache in Text, wäre lediglich ein textueller Fehler die Folge. Ebenso wäre eine Fehlerkennung in einem reinen Auskunftssystem nicht kostenkritisch. Auch bei der hier vorgestellten Applikation „Grafikeditor“ wären derartige Folgen gering, da lediglich am Monitor keine oder eine unerwünschte Aktion stattfindet. Die in diesem Absatz beschriebenen Applikationen sind also bezüglich ihrer Sicherheit völlig unkritisch.

Anders jedoch verhält es sich bei der ebenfalls untersuchten Domäne „Serviceroboter“. Bei einer Fehlerkennung wäre unter Umständen die ungewollte Zerstörung eines Objekts (z.B. durch zu festes Greifen oder durch Entsorgung in den Mülleimer) die Folge. Im Rahmen eines sprachverstehenden Buchungssystems muß eine eventuell ungewollte, kostenintensive Hotel- oder Flugbuchung aufgrund eines Erkennungsfehlers ausgeschlossen werden. Sollte eine Steuerung von Maschinen oder Fahrzeugen mit gesprochener Sprache erfolgen, ist ein Schaden an Mensch oder Maschine durch einen Unfall des Fahrzeugs unter allen erdenklichen Umständen zu vermeiden.

Es liegt auf der Hand, daß bei sicherheitskritischen Aspekten auf jeden Fall eine Rückbestätigung der gemachten Anweisung über einen 100% robusten Eingabekanal (Tastatur oder Knopfdruck) zu erfolgen hat. So muß der Bediener bei der „Serviceroboter“-Applikation nach der Intentionsdecodierung seine decodierte Anweisung mit Tastendruck bestätigen, bevor sie vom System ausgeführt wird [Hav96].

7.3 Dialogführung

Bei Applikationen ohne andere Eingabekanäle (z.B. Telefon ohne Mehrfrequenzwahlverfahren) können Erkennungsfehler durch eine geschickte Dialogführung lokalisiert und auf Wunsch korrigiert werden. Als einfachste Möglichkeit bietet sich hier die sprachliche Rückbestätigung mit „ja“ oder „nein“ an. Die umständliche und unnatürliche Rückbestätigung steigert zwar die Robustheit, wirkt sich jedoch nachteilig auf die Akzeptanz aus, da diese Art der Bestätigung bei zwischenmenschlicher Kommunikation fehlt.

System:	„Bitte die nächste Anweisung.“
Benutzer:	„Lösche die blaue Kugel!“
System:	„Wollen Sie die braune Kugel wirklich löschen? Antworten Sie nur mit ‚ja‘ oder ‚nein‘.“
Benutzer:	„Nein.“
System:	„Ich führe keine Aktion aus und warte auf die nächste Anweisung.“

Tab. 7.1: Dialogbeispiel zur Unterdrückung eines Erkennungsfehlers

Die explizite Bitte an den Benutzer, nur mit ja oder nein zu antworten, ist an dieser Stelle sinnvoll, da die sonst zu erwartende Antwort höchstwahrscheinlich „nein – die blaue“, (d.h. Ablehnung und Korrektur innerhalb derselben Äußerung) lauten würde. Dies würde den Aufwand der Decodierung an dieser Stelle erheblich erschweren.

Denkbar wäre an dieser Stelle auch der Fall einer korrekten Erkennung oder einer korrekten semantischen Decodierung, die deshalb nicht ausgeführt werden kann, da eine Mehrdeutigkeit vorliegt. Ein sinnvoller Klärungsdialog ermöglicht die erforderliche Disambiguisierung, bevor das Kommando auf dem Rechner ausgeführt wird.

System:	„Bitte die nächste Anweisung.“
Benutzer:	„Lösche den Zylinder!“
System:	„Es befinden sich zwei Zylinder auf dem Bildschirm. Welchen meinen Sie?“
Benutzer:	„Den weißen Zylinder.“
System:	„Wollen Sie den weißen Zylinder wirklich löschen? Antworten Sie nur mit ‚ja‘ oder ‚nein‘.“
Benutzer:	„Ja.“
System:	„Ihr Wunsch wird ausgeführt.“

Tab. 7.2: Dialogbeispiel zur Disambiguisierung einer korrekt decodierten Eingabe

Eine durch eine vorherrschende Mehrdeutigkeit erforderliche Objektspezifizierung – ähnlich obigem Dialogbeispiel in Tab. 7.2 – wurde für die Grafikeditor-Domäne implementiert. Dies erfolgt effizient durch eine situative Modifikation der im semantischen Modell befindlichen Wurzelwahrscheinlichkeiten f_0. Sollte die zu erwartende Spezifikation beispielsweise eine Formangabe und deren Nachfolger sein, so müßte in diesem Fall die Wurzelwahrscheinlichkeit des betreffenden Typs mit $f_0 = 1$ gewichtet werden. Die Wurzelwahrscheinlichkeiten aller nicht zu erwartender Typen bekämen den Wert $f_0 = 0$. Daraus resultieren je nach Dialogsituation verschiedene auf die jeweilige Situation angepaßte semantische Modelle [Kor96].

7.4 Dialog-Gestaltungsrichtlinien

Innerhalb weiterführender Arbeiten [Har93][Klo95] wurden Richtlinien für einen effizienten natürlichsprachlichen Mensch-Maschine-Dialog erarbeitet. Diese waren zwar aufgrund vorgegebener Rahmenbedingungen auf eine reine Telefonanwendung beschränkt, ihre Übertragung auf die realisierten Domänen „Grafikeditor“ und „Serviceroboter“ ist jedoch prinzipiell möglich. Durch eingehende Untersuchungen einer Vielzahl von sprachlichen Dialogen mit vielen verschiedenen Sprechern konnten Richtlinien zur Gestaltung sprachlicher Dialoge aufgestellt werden, um Benutzerzufriedenheit und damit Benutzerakzeptanz zu gewährleisten:

- **Aufbau des Dialogs**
  - bei Verfügbarkeit von anderen Eingabekanälen (z.B. Tastatur, Maus) eventuell einen Teil der Informationseingabe sinnvoll auslagern[31)],

31) Wie im Ausblick (siehe Kap. 11) angesprochen benötigen graphische Steuerungen – und dazu gehört zweifelsohne auch der im Rahmen dieser Arbeit implementierte sprachverstehende Grafikeditor – eine Koordinateneingabe, für die eine sprachliche Eingabe unzweckmäßig erscheint. Dieses Problem könnte mit multimodaler Eingabe (Sprechen und Zeigen) optimal gelöst werden.

  - maximal drei Ebenen, maximal vier Verzweigungen pro Ebene,
  - Funktionalität, Befehle und Programmstruktur explizit beschreiben,
  - Hinweis geben, daß Computer spricht, falls nicht offenkundig.
- **Ansagetexte**
  - Genau formulieren, Sachverhalte nicht umschreiben,
  - maximal drei Antworten vorgeben,
  - Fachausdrücke und ungewohnte Fremdwörter vermeiden,
  - die Zahl ‚2' als „zwo" ansagen[32)],
  - nach Möglichkeit keine codierten Antworten verlangen, ansonsten zuerst Text und dann Code[33)],
  - immer dieselbe Eingabeaufforderung verwenden
  - unterschiedliche Stimmen für Normalbetrieb, Hilfestellung und Fehlermeldung einsetzen.
- **Fehlerbehandlung und -vermeidung**
  - In den Expertenmodus mit „nein" verzweigen,
  - die erkannte bzw. decodierten Äußerung wiederholen,
  - auf jede Frage eine Antwort verlangen,
  - auf jede Aktion eine Bestätigung verlangen,
  - vorgegebene Antworten möglichst mehrsilbig vorsehen[34)],
  - keine ungerechtfertigten Annahmen,
  - Schuld an Fehlern immer dem Rechner zuweisen, Benutzer mit Respekt behandeln,
  - bei Problemen Hilfeansage abspielen, Hilfe kontextabhängig darbieten,
  - Fehler kontextabhängig melden[35)],
  - falls erforderlich, Frage zwei- bis dreimal wiederholen,
  - falls die wiederholte Fehlerkorrektur zu keinem positiven Ergebnis gelangt, Dialog in einen fest definierten Zustand lenken.
- **Ständig bereitzustellende Interaktionsmöglichkeiten**
  - Hilfeansage
  - letzte Ansage wiederholen
  - letzte Eingabe löschen
  - letzte Eingabe erneut eingeben
  - einen Schritt zurück gehen

---

32) Der Benutzer soll im Falle einer Eingabe motiviert werden, auch mit „zwo" zu antworten. Damit steigt die Erkennungssicherheit, da die Unterscheidbarkeit zwischen „zwo" und „drei" wesentlich höher ist als diejenige zwischen den akustisch sehr ähnlichen Worten „zwei" und „drei".

33) Dies ist unter Umständen unvermeidbar, wenn zur Informationseingabe ein reiner Zifferner-kenner zur Verfügung steht. Ein Beispiel einer in diesem Sinne korrekten Ansage wäre somit: „Für München sprechen Sie ‚eins', für Frankfurt ‚zwei' und für Hamburg ‚drei'."

34) Dadurch werden die Unterscheidbarkeit und die damit verbundene Erkennungssicherheit wesentlich erhöht.

35) Eine ungenügende Fehlermeldung wäre „Ein Fehler ist aufgetreten!", ohne die Fehlerursache zu spezifizieren.

- zum Programmanfang springen
- Programm beenden
- Lautstärke der Ansagen ändern

Psychologische Untersuchungen haben ergeben, daß durch einen geschickt geführten Dialog, in dem die zur Kommunikation notwenigen Schlüsselworte und Phrasen dem Benutzer vorgegeben werden, die Decodierungsleistung signifikant steigt. Die Variabilität möglicher Eingaben nimmt dabei spürbar ab, da ein Benutzer unbewußt die ihm „in den Mund gelegten“ Worte zur sprachlichen Informationseingabe wiederverwendet [Zol82] [Zol91]. Im folgenden Dialogbeispiel werden die bedeutungstragenden Worte „erzeugen“, „verändern“ und „löschen“ in der Systemmeldung erwähnt. Dadurch wird der Benutzer bewußt oder unbewußt zum Gebrauch dieser Worte animiert.

System:	„Bitte die nächste Anweisung.“
Benutzer:	„Na mach‘ doch mal den gelben Kegel da weg!“
System:	„Leider kann ich Ihre Eingabe nicht verstehen. Ich kann Objekte erzeugen, verändern oder löschen. Bitte wiederholen Sie Ihre Anweisung.“
Benutzer:	„Lösche den gelben Kegel.“
System:	„Wollen Sie den gelben Kegel wirklich löschen? Antworten Sie nur mit ‚ja‘ oder ‚nein‘.“
Benutzer:	„Ja.“
System:	„Ihr Wunsch wird ausgeführt.“

Tab. 7.3: Dialogbeispiel zur Vorgabe relevanter Befehlsworte

Grundsätzlich ist ein sprachlicher Mensch-Maschine-Dialog beim jetzigen Stand der Technik immer ein Kompromiß zwischen möglichst wenigen Einschränkungen (im Sinne von optimaler Benutzbarkeit bei geringem Lernaufwand) und möglichst starrem Dialogschema (im Sinne von robuster Erkennung durch fest vorgegebene Antwortmöglichkeiten). Diesen Kompromiß zumindest etwas in Richtung minimaler Einschränkungen zu verlassen, ist das Ziel der vorliegenden Arbeit.

Kapitel 8

Automatische semantikbasierte Sprachübersetzung

Die semantische Gliederung kann nicht nur als Ausgabe einer semantischen Decodierung betrachtet werden, sondern auch als Eingabe zur semantikbasierten Generierung natürlicher Sprache. Die aus einer zweistufigen Abfolge aus Wortkettengenerator und linguistischer Nachbearbeitung bestehende Sprachproduktion läuft nicht wie bei den üblichen Ansätzen [Bus93][Maa82] mittels eines linguistischen Regelwerks ab, sondern bedient sich mit entsprechenden Modellen für Wortwahl und Wortstellung stochastischer Methoden. Lediglich die innerhalb der linguistischen Nachbearbeitung vorgenommene korrekte Anpassung der Flexionen ist regelbasiert, wobei das grundlegende Vorgehen von klassischen linguistischen Ansätzen abweicht. Der hier vorgestellte Ansatz läuft daher primär unter (linguistisch betrachtet) pragmatischen Gesichtspunkten ab, so daß keinerlei kognitionswissenschaftliche Erkenntnisse, die mit der Generierung natürlicher Sprache in Zusammenhang stehen [Her93], betrachtet werden.

Am Beispiel der generierten Sprachen Deutsch, Englisch, Französisch und Slowenisch wird im Rahmen dieser Arbeit die multilinguale Verwendbarkeit der semantischen Gliederung aufgezeigt. Finden nun semantische Decodierung und Sprachproduktion in verschiedenen Sprachen statt, kann eine automatische, semantikbasierte Übersetzung [Mül96a] innerhalb der betrachteten Domäne realisiert werden.

8.1 Wortkettengenerator

Im Gegensatz zum Wortkettengenerator der semantischen Decodierung, welcher zu einer vorliegenden semantischen Gliederungshypothese viele Wortkettenhypothesen entstehen läßt, erzeugt der an dieser Stelle beschriebene Wortkettengenerator aus einer vorliegenden semantischen Gliederung S_E nur eine einzige korrespondierende Wortkette W_g. Dabei spielt es prinzipiell keine Rolle, ob die generierte Wortkette natürlichsprachlich ist oder nicht. Der stochastische Prozeß der Wortkettengenerierung mit der in [Sta95] [Sta97b] beschriebenen Grammatik kann als komplexes Zustands-Übergangs-Netzwerk, ähnlich einem hierarchischen Hidden-Markov-Modell [Rab89], aufgefaßt werden. Jedes

Semun korrespondiert zu einem syntaktischen Modul (SM), welches entsprechend der vorliegenden semantischen Gliederung mit anderen SMen zu einem hierarchischen syntaktischen Netzwerk verbunden ist. Innerhalb dieses Netzwerks beeinflussen die Übergänge zwischen einzelnen Zuständen die Wortstellung, die Emissionen von Worten aus bestimmten Zuständen die Wortwahl. Die Wahrscheinlichkeit $P(W|S_E)$ für eine bestimmte Wortkette W unter der Voraussetzung einer vorliegenden semantischen Gliederung S_E berechnet sich aus dem Produkt über alle Übergangswahrscheinlichkeiten a_n sowie über alle Emissionswahrscheinlichkeiten b_n und c_n entlang eines bestimmten Pfades durch das gesamte zur semantischen Gliederung S_E gehörende syntaktische Netzwerk (vgl. dazu auch Kap. 4.2.2 und Gl. (4.11)):

$$P(W|S_E) = \prod_{n=1}^{N} d_n = \prod_{n=1}^{N} (a_n \cdot b_n \cdot c_n) \tag{8.1}$$

Die zur Wortkettengenerierung benötigte Wissensbasis ist das syntaktische Modell, das, wie in den folgenden Unterpunkten beschrieben wird, sowohl stochastisch trainiert als auch regelbasiert eingesetzt werden kann.

8.1.1 Stochastisch trainierte syntaktische Modelle

Zur Wortkettenproduktion können die zur semantischen Decodierung trainierten syntaktischen Modelle übernommen werden. Durch deren generative Mächtigkeit sind sie nicht nur in der Lage, vorliegende Wortketten grammatikalisch abzuleiten und einer noch unbekannten semantischen Gliederung S zuzuordnen (d.h. zu parsen), sondern sie können auch die zu einer vorliegenden semantischen Gliederung S_E korrespondierenden Wortketten W erzeugen. Die Vielfalt dieser Wortketten W wird durch die Maximierung der Wahrscheinlichkeit auf eine einzige beschränkt, so daß als generierte Wortkette W_g diejenige mit der maximalen Wahrscheinlichkeit $P(W|S_E)$ ausgegeben wird [Mül95b]:

$$W_g = \operatorname*{argmax}_W P(W|S_E) \tag{8.2}$$

Bei der Verwendung zur Sprachproduktion sind die stochastisch trainierten, syntaktischen Modelle jedoch mit Nachteilen behaftet. Die in Kapitel 8.2 beschriebene linguistische Nachbearbeitung benötigt zur Flexionsanpassung ein fest vorgegebenes (Schlüssel-) Wort in einer fest vorgegebenen grammatikalischen Form, in der Regel Maskulinum, Nominativ, Singular. Dies ist jedoch, wie auch die Wortstellung innerhalb der Wortkette, bei den stochastisch trainierten Modellen nicht sicher gewährleistet. Deshalb wird zur Wortkettengenerierung die Verwendung regelbasierter syntaktischer Modelle dringend empfohlen. Dies gilt insbesondere in Verbindung mit der linguistischen Nachbearbeitung.

8.1.2 Regelbasierte syntaktische Modelle

Werden sämtliche Möglichkeiten innerhalb eines stochastisch trainierten Modells auf jeweils eine fest vorgegebene Entscheidung (mit „Wahrscheinlichkeit“ = 1) begrenzt, entartet das vormals stochastische Modell zum regelbasierten Modell. Die Struktur eines re-

gelbasierten syntaktischen Modells weicht prinzipiell nicht vom stochastischen Pendant ab. Vielmehr besteht der Unterschied darin, daß lediglich fest vorgegebene Entscheidungen mit den Wahrscheinlichkeiten 0 oder 1 vorzufinden sind, d.h. keinerlei Alternativen ermöglicht werden. Innerhalb jedes zu einem Semun $s_n \in S_E$ gehörenden SMs $A(s_n)$ werden folgende (regelbasierte) Festlegungen getroffen:

- Die Wortstellung und die Voraussetzung, ein bedeutungsloses Wort $w^-[s_n]$ zu emittieren, werden mit entsprechenden Übergangswahrscheinlichkeiten $a_{n,\mu,\nu}$, die nur einen der Werte Null oder Eins annehmen können, fest vorgegeben. Für die sich ergebende Pfadwahrscheinlichkeit eines Semuns s_n gilt daher stets $a_n = 1$.
- Zu einem vorliegenden Semun s_n kann nur ein bestimmtes bedeutungstragendes Wort $w^+[s_n]$ mit der Emissionswahrscheinlichkeit $b_n = 1$ emittiert werden.
- Falls für ein Semun s_n ein bedeutungsloses Wort $w^-[s_n]$ emittiert wird, ist auch dieses fest vorgegeben. Entsprechend Gl. (4.9) ergibt sich für die Emisisonwahrscheinlichkeit $c_n = 1$.

Die Modulwahrscheinlichkeit d_n jedes Semuns $s_n \in S_E$ errechnet sich aus dem Produkt der vorangegangenen Wahrscheinlichkeiten a_n, b_n und c_n:

$$d_n = a_n \cdot b_n \cdot c_n \overset{\text{nur für regelbasierte Modelle}}{=} 1 \tag{8.3}$$

Damit ist die Wortkette W_g bei gegebener semantischer Gliederung S_E fest vorgegeben und hat, S_E vorausgesetzt, folgende bedingte Wahrscheinlichkeit:

$$P(W_g | S_E) = \prod_{n=1}^{N} d_n \overset{\text{nur für regelbasierte Modelle}}{=} 1 \tag{8.4}$$

Natürlich entfällt für den regelbasierten Fall die in Gl. (8.2) geforderte Maximierung dieser bereits größtmöglichen Wahrscheinlichkeit. Die Wortkette W_g steht ja ohnehin mit gegebenem S_E bereits fest.

8.1.3 Prinzip der Wortketten-Erzeugung

Wie in obigen Abschnitten dargelegt liefert der Wortkettengenerator eine zur semantischen Gliederung S_E passende Wortkette W_g. In den folgenden Beispielen werde vereinfachend ein regelbasiertes syntaktisches Modell verwendet. Die Eingabe stellt dabei die in umseitiger Abbildung 8.1 gezeigte semantische Gliederung S_E dar.

Anhand der vorliegenden semantischen Gliederung S_E wird ein aus mehreren SMen $A(s_n)$ bestehendes syntaktisches Netzwerk gebildet, welches wegen des starren, regelbasierten syntaktischen Modells keine Alternativen zuläßt. Dieses Netzwerk wird beginnend vom Knoten beg(s_1) bis zum Knoten end(s_1) durchlaufen. Sofern ein Zustand $A(q_x[s_n])$ betreten wird, wird beim Knoten beg($q_x[s_n]$) fortgesetzt. Aus den entsprechenden Zuständen $B(s_n)$ bzw. $C(s_n)$ innerhalb eines SMs werden Worte emittiert. Hierbei spielt es jedoch

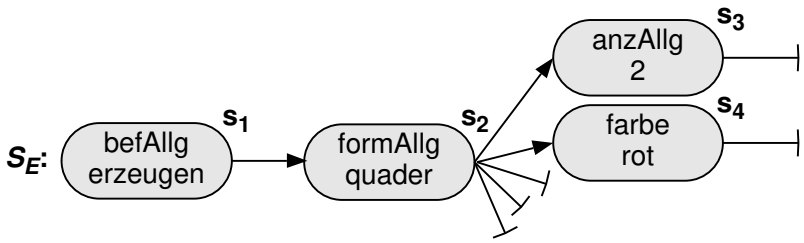


Abb. 8.1: Beispielhafte semantische Gliederung S_E als Eingabe zur Wortkettenproduktion

keine Rolle, inwiefern das emittierte Wort im Sinne der Definition der semantischen Gliederung wirklich bedeutungstragend oder bedeutungslos ist. Ob die somit produzierte Wortkette natürlichsprachlich ist oder nicht, und in welcher Sprache diese Wortkette ist[36)], ist für den prinzipiellen Ablauf des hier vorgestellten Verfahrens völlig unerheblich. Falls nötig, kann von einem Zustand $B(s_n)$ bzw. $C(s_n)$ auch eine Sequenz mehrerer zusammenhängender Worte emittiert werden.

Dabei ist zu beachten, daß im Gegensatz zur letztendlichen Wortwahl mit korrekter Flexion die Wortstellung von der nachfolgenden linguistischen Nachbearbeitung nicht mehr verändert werden kann. Damit wird die Wortstellung bereits durch die Wortkettengenerierung fest vorgegeben. Aus diesem Grund sind den festzulegenden Übergangswahr-

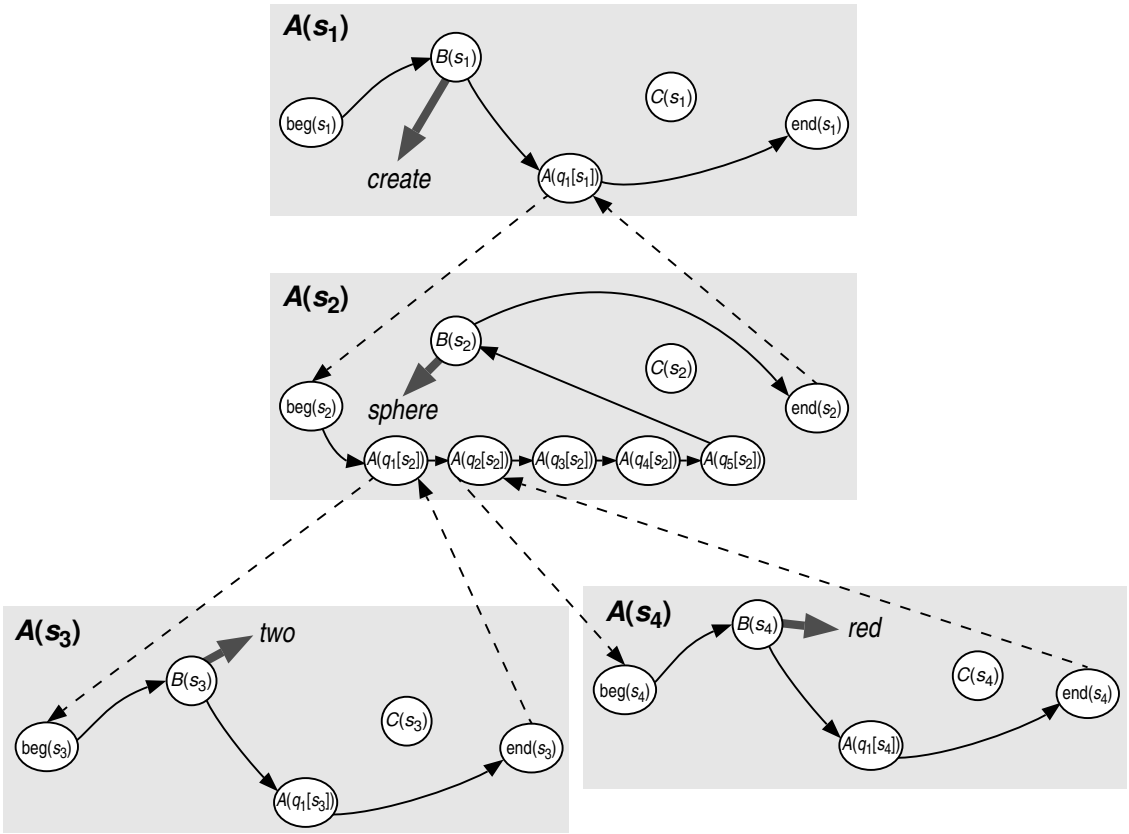


Abb. 8.2: Syntaktisches Netzwerk einer vorgegebenen semantischen Gliederung S_E zur Produktion der Wortkette $W_{g,\text{eng}}$ in englischer Sprache (Darstellung nach [Sta97b])

36) Siehe dazu auch Kapitel 8.4.

scheinlichkeiten $a_{n,\mu,\nu}$ innerhalb des regelbasierten syntaktischen Modells große Aufmerksamkeit zu widmen. Gegebenenfalls sind diese Wahrscheinlichkeiten iterativ zu korrigieren, um eine syntaktisch korrekte Wortfolge zu gewährleisten.

Als Beispiel sei in vorheriger Abbildung 8.2 ein zur semantischen Gliederung S_E gehörendes syntaktisches Netzwerk aufgespannt, welches zu der vorliegenden semantischen Gliederung S_E eine englischsprachige Wortkette $W_{g,\mathrm{eng}}$ generieren soll. Wird das vorliegende syntaktische Netzwerk vom Knoten beg(s_1) bis zum Knoten end(s_1) abgearbeitet, wird beim Durchschreiten des Pfades die Wortkette

$$W_{g,\mathrm{eng}}\text{: ‚create two red sphere‘}$$

emittiert. Da die semantische Gliederung keinerlei grammatikalische Information enthält und innerhalb des syntaktischen Modells Wortemissionen nur vom Semun selbst abhängen und nicht von Nachbar-Semunen beeinflußt werden, besteht hier keinerlei Wissen darüber, ob das Wort „sphere“ im Singular oder Plural zu erscheinen hat.

Zur Verdeutlichung sei nun dieselbe semantische Gliederung S_E als Basis zur Produktion einer französischsprachigen Wortkette verwendet. In folgender Abb. 8.3 wird das entsprechende syntaktische Netzwerk mit dem durchlaufenen Pfad gezeigt.

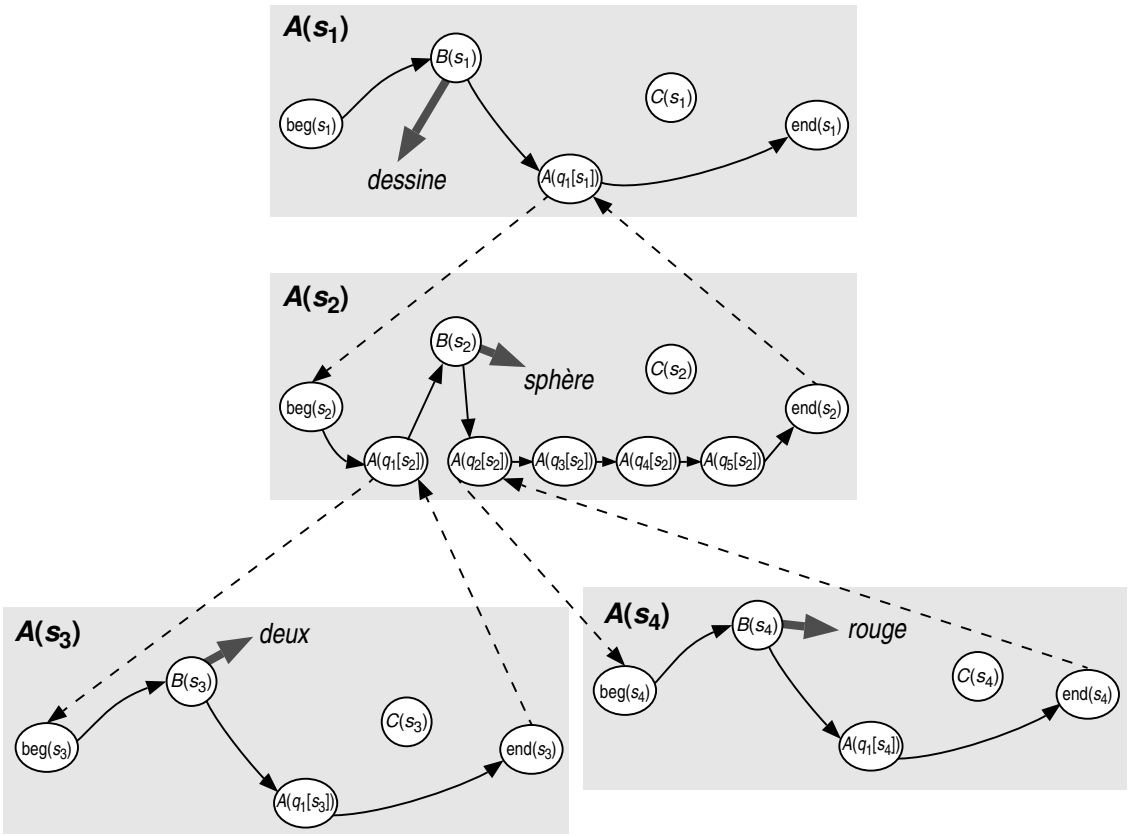


Abb. 8.3: Syntaktisches Netzwerk einer vorgegebenen semantischen Gliederung S_E zur Produktion der Wortkette $W_{g,\mathrm{frz}}$ in französischer Sprache (Darstellung nach [Sta97b])

Während des Durchlaufes emittiert dieses syntaktische Netzwerk aufgrund des verwendeten syntaktischen Modells die französische Wortkette

$$W_{g,\mathrm{frz}}\text{: ‚dessine deux sphère rouge‘.}$$

Analog zur oben beschriebenen englischsprachigen Generierung besteht auch in diesem Beispiel keinerlei Wissen darüber, ob in diesem durch S_E festgelegten Zusammenhang die französischen Worte „sphère“ („kugel“) und „rouge“ („rot“) im Singular oder Plural erscheinen müssen. Dabei ist der Unterschied im SM $A(s_2)$ zwischen der englisch- und der französischen Version zu beachten. Damit im Französischen das formbezeichnende Wort vor dem farbbezeichnenden Wort emittiert werden kann, müssen die Zustände in einer anderen Abfolge als im englischsprachigen Beispiel durchlaufen werden.

Der auf der Basis der Viterbi-Suche [Vit73] funktionierende Algorithmus des Wortkettengenerators weicht prinzipiell kaum von dem zur semantischen Decodierung implementierten Gegenstück ab. Eine ausführliche Beschreibung findet sich dazu in [Sta97b].

8.2 Linguistische Nachbearbeitung

Wie bereits im vorigen Kapitel veranschaulicht wurde, können, da die semantische Gliederung keinerlei grammatikalische Information enthält, Genus, Kasus bzw. Numerus mancher Worte aus der generierten Wortkette W_g falsch sein. Dennoch kann der semantische Inhalt dieser syntaktisch mitunter unkorrekten Wortkette von einem menschlichen Informationsempfänger verstanden werden. So kann beispielsweise die Wortkette

$$W_{g,\mathrm{frz}}\text{: ‚dessine deux sphère rouge‘.}$$

vom einem mit einem französischsprachigen syntaktischen Modell ausgestatteten Wortkettengenerator anstatt der grammatikalisch korrekten Wortkette ‚dessine deux sphères rouges‘ ausgegeben werden. In diesem Fall waren entweder die jeweiligen Emissionswahrscheinlichkeiten der Singularworte ‚sphère‘ und ‚rouge‘ höher als diejenigen der korrekten Pluralworte ‚sphères‘ und ‚rouges‘. Es könnten jedoch auch eigens zur Sprachproduktion einsetzbare, regelbasierte syntaktische Modelle verwendet worden sein, die nur eine mögliche Emission zuslassen. In jedem Fall muß eine nachfolgende Instanz die auftretenden grammatikalischen Inkorrektheiten beseitigen. Eine generierte Wortkette W_g muß also linguistisch nachbehandelt werden.

Ein wesentlicher Vorteil des im folgenden vorgestellten Verfahrens ist, daß zur Nachbearbeitung nicht nur die Wortkette W_g, sondern auch die semantische Gliederung S_E zur Verfügung steht. Aus verarbeitungstechnischen Gründen wird nun nicht die Wortkette, sondern primär die semantische Gliederung S_E ausgehend von deren Wurzel (d.h. dem Semun s_1) bis hin zu deren Blättern (d.h. allen leeren Nachfolgern) rekursiv abgearbeitet. Wenn mindestens eines der zu einem Semun s_n gehörenden Worte $w^+[s_n]$ bzw. $w^-[s_n]$ ein Substantiv, Adjektiv oder Artikel ist, muß die den jeweils zuzuordnenden grammatikalischen Eigenschaften (d.h. Genus, Kasus, Numerus) entsprechende Flexion (Endung) die-

ses Wortes anhand des vorliegenden Flexionsmodells gefunden und dem zu verbessernden Wort angehängt werden.

Die grammatikalischen Eigenschaften Genus $\gamma[s_n]$, Kasus $\kappa[s_n]$ und Numerus $\eta[s_n]$ werden dabei mittels sprachspezifischer Grammatikregeln (siehe Kapitel 8.2.1) innerhalb der semantischen Gliederung S_E bestimmt und zunächst dem jeweiligen Semun s_n zugeordnet. Nur die Genusbestimmung wird unmittelbar oder indirekt von einem emittierten Substantiv bedingt, alle anderen Eigenschaften gründen ausschließlich auf der semantischen Gliederung. Dies ist natürlich im Vergleich zur klassischen Linguistik, welche syntaktische und semantische Bindungen und Beschreibungen nur aufgrund der Worte und Sätze vornimmt, ein vollständig verschiedener und neuartiger Ansatz! Die Eigenschaften $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ können dabei auf verschiedene Arten gebildet werden [Kai95a]:

Eine grammatikalische Eigenschaft eines zu untersuchenden Semuns s_n kann bestimmt werden abhängig

1. vom Typ $t[s_n]$ und vom Wert $v[s_n]$ bzw. nur bei der Genusbestimmung vom bedeutungstragenden Wort $w^+[s_n]$
   (Regel *SELBST*),
2. vom Typ und vom Wert des Vorgänger-Semuns
   (Regel *VOR_ÜBER*) oder
3. vom Typ $t[q_x[s_n]]$ und vom Wert $v[q_x[s_n]]$ des x-ten Nachfolger-Semuns
   (Regel *NACH_ÜBER*).

Die grammatikalische Eigenschaft kann auch übernommen werden

4. vom Vorgänger-Semun unabhängig von dessen Typ und Wert
   (Regel *VOR_EIGEN*),
5. vom x-ten Nachfolger-Semun $q_x[s_n]$ unabhängig von dessen Typ und Wert
   (Regel *NACH_EIGEN*) oder
6. von einem in S_E enthaltenen Semun, welches das aktuelle Objekt bezeichnet, unabhängig von dessen Typ und Wert
   (Regel *OBJEKT*).

Welche Regel angewendet wird, ist primär vom Typ und nur in Ausnahmefällen auch vom Wert des Semuns abhängig. Dabei ist zu beachten, daß die Festlegung von $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ für ein Semun s_n unabhängig voneinander mittels verschiedener Regeln (siehe obige Punkte 1. – 6.) erfolgen kann.

Die erlaubten Werte für $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ können für verschiedene zu generierende Sprachen variieren. Da der Sinn der linguistischen Nachbearbeitung lediglich die korrekte Flexionszuweisung und keine grammatikalische Analyse ist, werden $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ nicht im Sinne einer formalen sprachlichen Grammatik verwendet, sondern rein pragmatisch als Bezeichnung für die Flexionszuweisung. Daher können die hier verwendeten Bezeichnungen für $\gamma[s_n]$, $\kappa[s_n]$ und $\nu[s_n]$ von realen Geni, Kasi und Numeri abweichen. Sie werden in folgender Tabelle für die vier untersuchten Zielsprachen Deutsch, Englisch, Französisch und Slowenisch aufgelistet.

	Deutsch	**Englisch**	**Französisch**	**Slowenisch**
Genus $\gamma[s_n]$	Maskulinum Femininum Neutrum	Neutrum	Maskulinum Femininum	Maskulinum Femininum Neutrum
Kasus $\kappa[s_n]$	Nominativ Genitiv Dativ Akkusativ	AlleFälle Genitiv	NominAkkusativ Genitiv Dativ	Nominativ Genitiv Dativ Akkusativ Lokativ Instrumental
Numerus $\eta[s_n]$	Singular_Allgemein Plural_Allgemein Singular_Bestimmt Plural_Bestimmt Singular_Unbestimmt	Singular Plural	Singular Plural	Singular_Allgemein Dual_Allgemein Plural_Allgemein Singular_Bestimmt Dual_Bestimmt Plural_Bestimmt

Tab. 8.1: Vorrat an Geni, Kasi und Numeri für verschiedene Zielsprachen

Nachdem $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ für das zu untersuchende Semun $s_n \in S_E$ bestimmt wurden, werden diese Eigenschaften, soweit erforderlich, auf die dem Semun s_n zugeordneten Worte $w^+[s_n]$ und, falls vorhanden, $w^-[s_n]$ übertragen. Die grammatisch korrekte Endung wird dabei aus einem Flexionsmodell bezogen, welches die jeweilige Endung eines Wortes abhängig von $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ enthält (siehe dazu Kapitel 8.2.2).

8.2.1 Grammatik-Regeln

In dieser regelbasierten Wissenbasis werden die Regeln zur Bestimmung von Genus $\gamma[s_n]$, Kasus $\kappa[s_n]$ und Numerus $\eta[s_n]$ eines zu untersuchenden Semuns s_n in Abhängigkeit von dessen Aufbau (d.h. dessen Typ und Wert) sowie dessen Umgebung (d.h. dessen Vorgänger- und Nachfolgerkonstellation) bereitgestellt. Dabei ist zu beachten, daß die hierbei verwendeten Regeln sowohl für die verschiedenen Eigenschaften $\gamma[s_n]$, $\kappa[s_n]$, $\eta[s_n]$ als auch für verschiedene Sprachen signifikant voneinander abweichen. Für jede Sprache werden deshalb zur Bestimmung von $\gamma[s_n]$, $\kappa[s_n]$ und $\eta[s_n]$ drei voneinander getrennte Wissensbasen, deren Struktur jedoch gleich ist, eingelesen.

Die in Kap. 12.8 aufgelistete Wissensbasis mit Grammatikregeln zur Genusbestimmung verdeutlicht das zumeist auf der semantischen Gliederung basierende Vorgehen. Die in der ‚Grafikeditor'-Domäne verwendeten Typen werden in Typgruppen (z.B. *ANZAHL*, *AUSRI*, *FARBE* usw.) zusammengefaßt. Nun wird die Umgebung auf das Vorhandensein der weiterhin angegebenen Typgruppen untersucht. Sofern eine Typgruppe zutrifft, wird die vor der Typgruppe vermerkte Regel (z.B. *SELBST*, *VOR_ÜBER*, *NACH_EIGEN* usw.) zur Bestimmung der gesuchten Eigenschaft eingesetzt. Im Falle mehrer Kombinationsmöglichkeiten „gewinnt" diejenige Regel, welche als letzte in den jeweils verwendeten Grammatikregeln vermerkt ist.

8.2.2 Flexionsmodell

In dieser regelbasierten Wissenbasis sind vermerkt:

- Die nachzubearbeitenden Worte (d.h. Substantive, Adjektive oder Artikel) in identischer Schreibweise, wie sie vom syntaktischen Modell des vorgeschalteten Wortkettengenerators emittiert werden,
- im Falle eines Substantives dessen Genus $\gamma[s_n]$,
- im Falle eines Substantives sämtliche zu erwartende Flexionen in Form einer zweidimensionalen Matrix in Abhängigkeit von Kasus $\kappa[s_n]$ und Numerus $\eta[s_n]$,
- im Falle eines Adjektives oder Artikels sämtliche zu erwartende Flexionen bzw. in Ausnahmefällen ganze Wortvariationen in Form einer dreidimensionalen Matrix in Abhängigkeit von Genus $\gamma[s_n]$, Kasus $\kappa[s_n]$ und Numerus $\eta[s_n]$.

Als Beispiel für ein Flexionsmodell wird im Anhang ein Flexionsmodell für deutsche Sprache aufgelistet – siehe dazu Kap. 12.5.

Aus den beiden in den vorangegangenen Kapitel beschriebenen Modulen „Wortkettengenerator“ und „linguistische Nachbearbeitung“ wurde nach folgendem Blockschaltbild eine Produktion von Wortketten innerhalb der ‚Grafikeditor‘-Domäne implementiert.

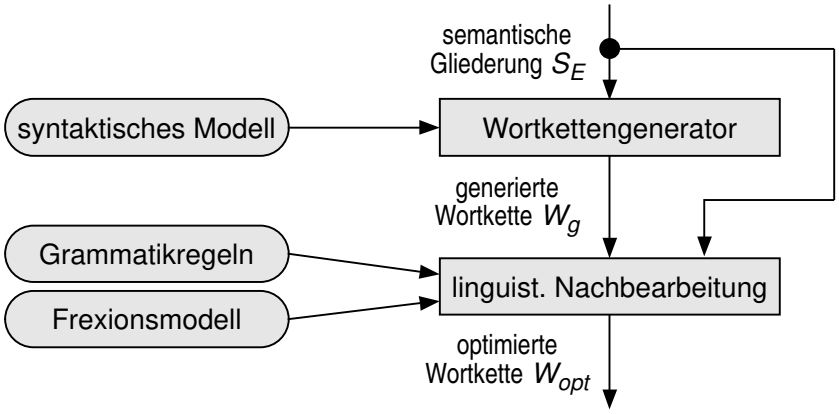


Abb. 8.4: Prinzip der Sprachproduktion auf der Basis der semantischen Gliederung

8.3 Ergebnisse

Die Sprachproduktion sollte innerhalb der ‚Grafikeditor‘-Domäne aus 307 realen semantischen Gliederungen[37)] korrespondierende Wortketten zu den vier Sprachen Deutsch, Englisch, Französisch und Slowenisch produzieren. Da Semantik und Syntax nicht von einer Maschine zu beurteilen sind, wurden mehrere Versuchspersonen gefragt, ob

37) Es wurden aus dem 1843 semantische Gliederungen umfassenden Trainingsmaterial, welches aus einer „Wizard of Oz“-Simulation [Mül95c] gewonnen wurde, 307 Gliederungen derart extrahiert, daß der Anteil jedes Sprechers ungefähr gleich blieb. Daher stammen die in diesem Abschnitt verwendeten semantischen Gliederungen von allen 33 Versuchspersonen.

- die Semantik einer Wortkette W_{opt} als verständlich oder unverständlich sowie
- die Syntax einer Wortkette W_{opt} als korrekt, unüblich oder falsch

empfunden wird. Die prozentualen Resultate dieser Befragung sind in folgender Tabelle 8.2 nach Sprachen geordnet zusammengestellt. Das dort ersichtliche, schlechtere Semantik-Ergebnis für Slowenisch ist mit dem Umstand dieser Sprache zu erklären, daß Mengenangaben nicht nur über explizite Zahlwörter, sondern auch implizit über Flexionen codiert werden können. Dies führte jedoch bei dem zur Sprachproduktion einheitlich gewählten Typen- und Werteinventar, welches auf die deutsche Sprache optimiert wurde, zwangsläufig zu Problemen (siehe dazu auch Kap. 8.4).

Zielsprache	**Semantik**	**Syntax**	
	verständlich	**korrekt**	**unüblich**
Deutsch	95,9 %	84,4 %	5,2 %
Englisch	94,8 %	81,1 %	10,4 %
Französisch	93,8 %	82,4 %	9,8 %
Slowenisch	88,3 %	82,1 %	8,5 %
Durchschnitt	**93,2 %**	**82,5 %**	**8,5 %**

Tab. 8.2: Ergebnisse der Sprachproduktion

Obige Ergebnisse mit verständlicher Semantik von durchschnittlich 93,2 % und einer nicht-falschen Syntax von durchschnittlich 91,0 % bestätigen, daß das in vorliegendem Abschnitt vorgestellte Verfahren zur Produktion einzelner Wortketten durchaus mit anderen gängigen, jedoch wesentlich aufwendigeren Verfahren [Bus93] konkurrieren kann, sofern sich die sprachlich zu generierende Domäne in klaren Grenzen bewegt.

8.4 Architektur des Übersetzungssystems

Die zunächst getroffene Annahme, daß die semantische Gliederung als wortnahe Repräsentationsebene des Bedeutungsinhaltes einer Äußerung multilingual (d.h. für mehrere Sprachen einsetzbar) ist, stellte sich im Rahmen der gemachten Untersuchungen als absolut richtig heraus. Äußerungen in germanisch-romanischen Sprachen Deutsch, Englisch und Französisch können sogar mit identischen semantischen Gliederungen repräsentiert werden. Für die slawische Sprache Slowenisch waren die Mengenangaben[38)], die nicht immer durch explizite Zahlworte, sondern auch implizit durch Flexionen wiedergegeben werden können, ein großes Problem. Während zur Sprachproduktion keinerlei Veränderungen nötig waren, mußten jedoch zur semantischen Decodierung leichte Modifikationen an verwendeten Typen und Werten vorgenommen werden, um die Erkennungsraten zu verbessern [Kai95a]. Auch eine Verwendung der semantischen Gliederung innerhalb fernöstlicher Sprachen (z.B. Japanisch, Chinesisch) erscheint prinzipiell bei geeigneter Anpassung des Typen- und Wertevorrates denkbar.

38) Im Slowenischen existieren drei unterschiedliche Numeri: Singular (Einzahl), Dual („Zweizahl“) und Plural (Mehrzahl).

Werden nun Vorverarbeitung, semantische Decodierung, Wortkettengenerierung, linguistische Nachverarbeitung und ein Sprachsynthesemodul (wie in Abb. 8.5 gezeigt) sequentiell verknüpft, kann ein lauffähiges System zur automatischen Übersetzung gesprochener oder geschriebener Sprache realisiert werden, sofern semantische Decodierung und Sprachproduktion in verschiedenen Sprachen – d.h. Quellsprache und Zielsprache – arbeiten [Mül96a]. Die jeweiligen Wissensbasen müssen sprachspezifisch in der jeweils gewünschten Quell- oder Zielsprache verfaßt sein.[39)]

- **Semantische Decodierung:**
  Semantisches, syntaktisches, akustisches, phonetisches Modell in Quellsprache
- **Wortkettengenerierung**
  Regelbasiertes syntaktisches Modell in Zielsprache
- **Linguistische Nachbearbeitung**
  Regelbasierte Grammatikregeln und regelbasiertes Flexionsmodell in Zielsprache

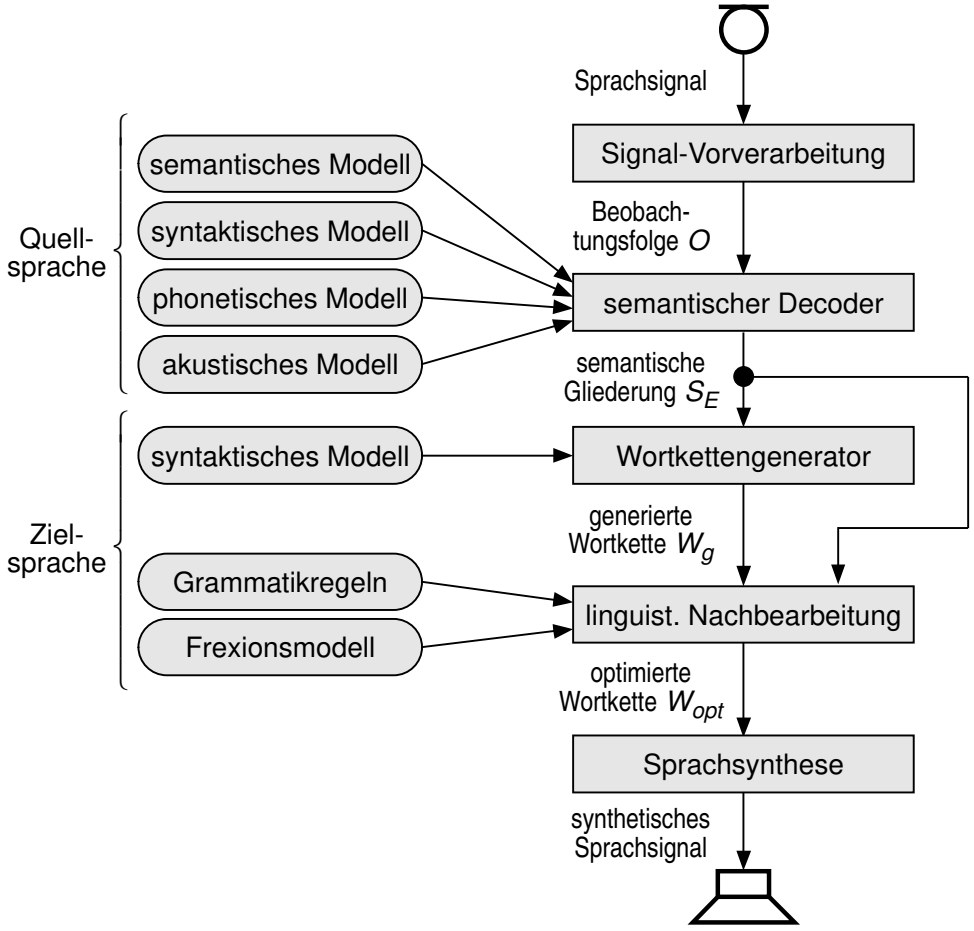


Abb. 8.5: Prinzip der semantikbasierten Sprachübersetzung als Kombination aus semantischer Decodierung und Sprachproduktion

39) Durch eine konsequent verfolgte Implementierungstechnik, sämtliches sprach- und domänenspezifisches Wissen in externen Dateien zu speichern, werden völlig sprach- und domänenunabhängige Algorithmen gewährleistet.

Ein denkbarer Einsatz für identische Quellsprache und Zielsprache wäre die Synonymproduktion (d.h. die Produktion gleichbedeutender Wortketten), die beispielsweise mit zusätzlichen Worten und abgewandelten grammatikalischen Eigenschaften in effizienten natürlichsprachlichen Dialogen Verwendung finden könnte.

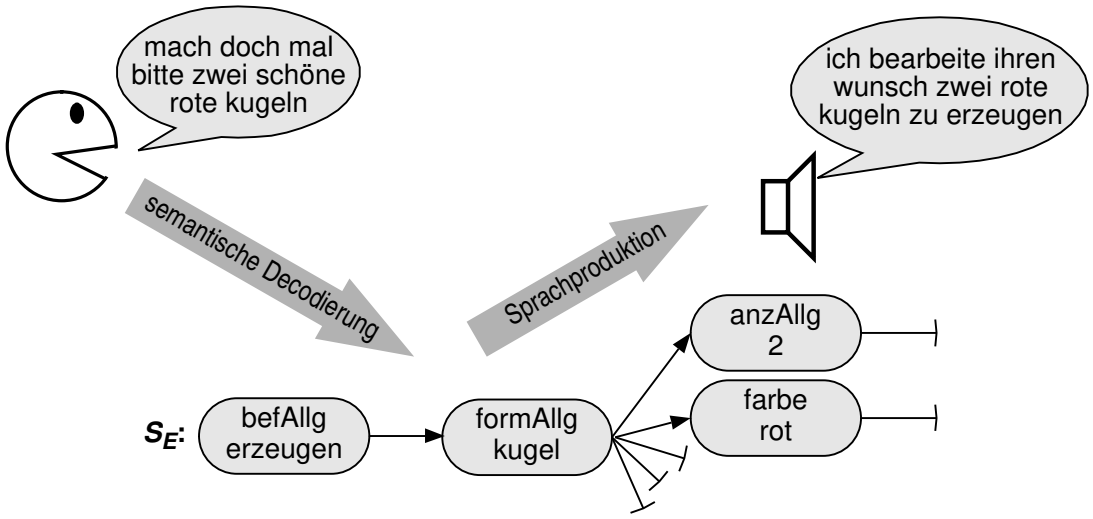


Abb. 8.6: Beispiel für einen Ausgabedialog auf der Basis der semantischen Gliederung S_E, die durch semantische Decodierung der Eingabeäußerung ermittelt wird

Das in diesem Abschnitt beschriebene Übersetzungssystem ist mit systembedingten Nachteilen verbunden. Bezüglich Domäne und Satzbau ist es den Beschränkungen des semantischen Decoders unterworfen. Spontansprachliche Effekte bilden wie bei anderen natürlichsprachlichen Systemen eine unumgängliche Fehlerquelle. Da kein großer zusammenhängender Text übersetzt werden soll, sondern nur kurze Äußerungen, erübrigt sich eine Analyse von Interpunktionszeichen, welche in [Loh82] eingesetzt wird. Der beschriebene Ansatz bedient sich auch keiner bewährten psycholinguistischen Übersetzungstheorie, wie zum Beispiel dem integrativen Ansatz der Skopustheorie [Sch91].

Es konnte jedoch der Beweis erbracht werden, daß ohne linguistisches Spezialwissen mittels der semantischen Gliederung und geeigneten Wissensbasen ein System zur domänenspezifischen Sprachübersetzung implementierbar ist, welches im Vergleich zu anderen Übersetzungssystemen [Bru82][Sch91] einen relativ einfachen und übersichtlichen Aufbau besitzt und sich durch geeignete Modifizierung der verwendeten Wissensbasen problemlos auf andere Domänen oder andere Sprachen portieren läßt.

Eine Einschränkung auf eine fest vorgegebene Domäne ist ebenfalls beim deutschen Verbundprojekt VERBMOBIL[40)] [Aue96][Wah93] vorgegeben, welches sich die Übersetzung gesprochener deutscher, englischer und japanischer Terminvereinbarungen zum Ziel gesetzt hat. An dieser Stelle sei angemerkt, daß ein Einsatz der in diesem Kapitel vorgestellten Übersetzerarchitektur innerhalb der VERBMOBIL-Domäne bei geeigneter Modifikation der verwendeten Wissensbasen grundsätzlich denkbar erscheint.

40) Auch am Lehrstuhl für Mensch-Maschine-Kommunikation der Technischen Universität München existiert eine Forschungsgruppe innerhalb des VERBMOBIL-Projekts.

Kapitel 9

Vorhersage von „Out of Vocabulary“-Rate und Vokabulargröße

Um die Parameter stochastischer Wissensbasen für sprachverarbeitende Applikationen zu ermitteln, wird Trainingsmaterial benötigt. Dafür muß ein Sprachkorpus bestehend aus einer Vielzahl von gesprochenen Äußerungen gesammelt werden. Diese dürfen zwar von mehreren Sprechern stammen, sollten jedoch innerhalb einer gleichbleibenden, definierten Domäne[41)] unter gleichbleibenden, definierten Bedingungen[42)] gewonnen worden. Zum Training von Wissensbasen, die Eigenschaften und Bindungen innerhalb der Wortebene beschreiben (z.B. syntaktisches Modell, Sprachmodell, Vokabular), wird ein aus vielen Wortketten bestehender Textkorpus herangezogen, wozu alle gesammelten gesprochenen Äußerungen nachträglich in Wortketten überführt werden müssen. Würde der Korpus von vorneherein schriftlich gesammelt werden, wären viele Eigenheiten gesprochener Sprache (z.B. spontansprachliche Effekte, Höflichkeitsfloskeln, besondere Wortwahl) nicht berücksichtigt.

Das Auftreten von unbekannten Worten innerhalb einer spracherkennenden Applikation wirkt sich auf die Benutzerakzeptanz äußerst schädlich aus, da es in der Regel mindestens einen Erkennungsfehler[43)] verursacht. Aktuelle Untersuchungen zeigen, daß im Mittel jedes unbekannte Wort im Mittel 1,2 [Cha95][Gau94] bis 1,6 [Woo95] Erkennungsfehler verursacht, welche vermieden werden könnten, wenn das betreffende Wort bekannt gewesen wäre. Der späteren Vermeidung von unbekannten Worten sollte also beim Erstellen bzw. Sammeln des Vokabulars größte Aufmerksamkeit eingeräumt werden! Im Gegensatz zu anderen durchgeführten Untersuchungen zur Minimierung der „Out of Vocabulary“-Rate (OOV-Rate) von sehr großen Korpora [Fet95][Ros95] wird in diesem

41) z.B. Grafikeditor, Telefonbedienung, Fahrplanauskunft, touristische Informationen, usw.

42) z.B. gesprochene „Wizard of Oz“-Simulation, Sammlung von getippten Anweisungen, kundiger bzw. unkundiger Benutzer, usw.

43) Ein Erkennungsfehler ist nicht nur ein falsches (d.h. in Sinne des Vokabulars anderes) Wort, sondern auch die Auslassung eines gesprochenen Wortes oder das Einfügen eines nicht gesprochenen Wortes.

Kapitel ein relativ kleiner domänenspezifischer Textkorpus betrachtet, dessen Vokabular nur aus einigen hundert Worten besteht.

Es ist sicherlich von Bedeutung, inwiefern ein vorliegender Textkorpus für wortrelevante Wissensbasen, insbesondere das Vokabular, ausreicht. Das Vokabular kann ja nur diejenigen Worte enthalten, die mindestens einmal im Trainingsmaterial auftreten. In diesem Zusammenhang sind insbesondere die folgenden Fragen interessant [Mül96b]:

- Wie groß ist die Wahrscheinlichkeit, daß mindestens ein Wort einer zu erwartenden Wortkette[44)] nicht im bestehenden Vokabular vorkommt? Diese Wahrscheinlichkeit wird im folgenden als „Out of Vocabulary“-Rate (OOV-Rate) des dem Vokabular zugrundeliegenden Korpus‘ bezeichnet.
- Kann eine Aussage betreffs der „Sättigung“ eines Korpus‘ gemacht werden? (Sättigung bedeutet in diesem Sinne, daß sich eine weitere Vergrößerung des Korpus‘ nicht mehr auf das resultierende Vokabular auswirkt.)
- Ist es überhaupt möglich, bei einem relativ kleinen Korpus vorherzusagen, wie sich eine Korpusvergrößerung (d.h. Hinzufügen weiterer Wortketten) auf die Vokabulargröße und die OOV-Rate auswirken wird?
- Wenn ja, wieviele Wortketten müßten dem Korpus (unter gleichen definierten Bedingungen) hinzugefügt werden, damit sich die zu erwartende OOV-Rate unter einen bestimmten Wert bewegt?
- Wieviele Worte würde das resultierende Vokabular nach dieser Korpusvergrößerung enthalten?

Die grundlegende Idee ist nun, den gegebenen Textkorpus in zwei Teilkorpora aufzuspalten: Ein abgespaltener Testkorpus, der M Wortketten des ursprünglichen Korpus‘ enthält, und einen verbleibenden Subkorpus, der einen um genau diese M Wortketten verminderten Korpus repräsentiert und aus dem ein zugehöriges Teilvokabular gebildet wird. Es wird festgestellt, wieviele Worte das Teilvokabular enthält und mit welcher Wahrscheinlichkeit eine Wortkette aus dem Testkorpus in diesem Teilvokabular, welches eben nicht aus diesen Wortketten gebildet wurde, enthalten ist. Werden beide Größen nun für alle denkbaren Subkorpus- und Testkorpus-Kombinationen berechnet und gemittelt, wobei die jeweilige Anzahl der im Sub- und Testkorpus enthaltenen Wortketten unverändert bleibt, ergibt sich eine mittlere Vokabulargröße und eine mittlere OOV-Rate genau für diese Testkorpusgröße M. Nun werden letztere gemittelte Größen für alle sinnvollen M bestimmt. Es ergeben sich von der Testkorpusgröße abhängige Werte der mittleren Vokabulargröße und der mittleren OOV-Rate. Mittels analytischer Funktionen, welche diese Verläufe möglichst gut approximieren, lassen sich für das hypothetische „Wegnehmen einer negativen Zahl von Wortketten“ (das bedeutet für das Hinzufügen von Wortketten) mittlere Vokabulargröße und mittlere OOV-Rate vorhersagen.

---

44) Eine zu erwartende Wortkette sollte innerhalb derselben Domäne und unter denselben Bedingungen gewonnen werden wie diejenigen Wortketten des ursprünglichen Korpus‘, aus welchem das bestehende Vokabular gebildet wurde.

9.1 Korpusparameter

Bei den hier angestellten Betrachtungen wird ein Textkorpus K zugrundegelegt, welcher aus L Wortketten W_l mit $1 \leq l \leq L$ besteht:

$$K = \{W_1, W_2, \ldots, W_l, \ldots, W_L\} \tag{9.1}$$

Jede Wortkette W_l besteht aus einem oder mehreren Worten und repräsentiert eine zu erwartende Benutzer-Äußerung innerhalb der betrachteten Domäne. Die Korpusgröße $|K|$ wird durch die Anzahl der Wortketten beschrieben:

$$|K| = L \tag{9.2}$$

Das Vokabular $V(K)$ ist diejenige Menge von T Worten, die mindestens einmal im Korpus K vorkommen. Mehrfach vorkommende Worte werden jedoch nur einmal in diese Menge aufgenommen.

$$V(K) = \{w_1, w_2, \ldots, w_t, \ldots, w_T\} \tag{9.3}$$

Die Vokabulargröße $|V(K)|$ ist in diesem Fall die Anzahl aller mindestens einmal vorkommender Worte im Korpus K:

$$|V(K)| = T \tag{9.4}$$

9.2 Bestimmung der OOV-Rate eines beliebigen Subkorpus

Der vorgegebene Korpus K wird in zwei Teilkorpora aufgespalten, in einen verbleibenden Subkorpus K_{sub} und in einen Testkorpus K_{test}. Dabei gilt:

$$K = K_{\text{sub}} \cup K_{\text{test}} \tag{9.5}$$

Der Testkorpus besteht dabei aus einer beliebigen Kombination von M Wortketten $W_{l_1}, \ldots, W_{l_M}$, die im ursprünglichen Korpus K vorkommen. Es gilt:

$$K_{\text{sub}} = K \setminus \{W_{l_1}, \ldots, W_{l_M}\} \text{ und } \left|K_{\text{sub}}\right| = L - M \tag{9.6}$$

$$K_{\text{test}} = \{W_{l_1}, \ldots, W_{l_M}\} \text{ und } \left|K_{\text{test}}\right| = M \tag{9.7}$$

Dabei gilt $l_1 \neq l_2 \neq \ldots \neq l_M$ mit $1 \leq l_m \leq L$ sowie $1 \leq m \leq M$ und $1 \leq M \leq L - 1$.

Für das aus dem Subkorpus gebildete Vokabular $V(K_{\text{sub}})$ gilt:

$$V(K_{\text{sub}}) \subseteq V(K) \tag{9.8}$$

Für die Vokabulargröße $\left|V(K_{\text{sub}})\right|$ kann folgende Aussage getroffen werden:

$$\left|V(K_{\text{sub}})\right| \leq \left|V(K)\right| \tag{9.9}$$

Das Vokabular verkleinert sich genau dann, wenn ein oder mehrere Worte von $V(K)$ ausschließlich in K_{test} und damit nicht in K_{sub} vorkommen.

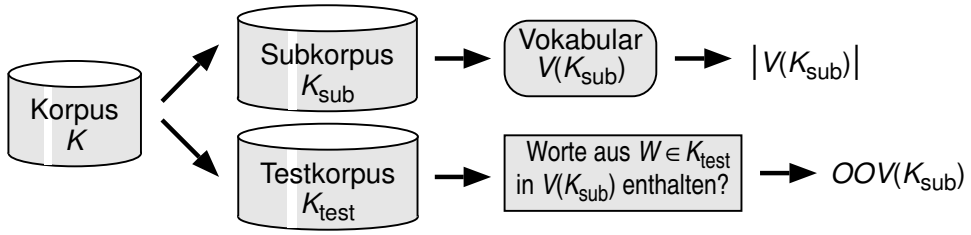


Abb. 9.1: Aufteilung eines Korpus‘ in einen Subkorpus und einen Testkorpus sowie Extraktion der relevanten Größen $|V(K_{\text{sub}})|$ und $OOV(K_{\text{sub}})$

Durch die Aufspaltung in die beiden Korpora K_{sub} und K_{test} ist es möglich, eine Wahrscheinlichkeit zu bestimmen, mit der mindestens ein Wort einer beliebigen Wortkette aus K_{test} nicht im Vokabular $V(K_{\text{sub}})$ enthalten ist. Diese Wahrscheinlichkeit wird im folgenden als OOV-Rate $OOV(K_{\text{sub}})$ bezeichnet. Sie stellt das Verhältnis zwischen der Häufigkeit von „OOV-Wortketten“ und der Größe des Testkorpus‘ dar:

$$OOV(K_{\text{sub}}) = \frac{\sum_{m=1}^{M} \text{oov}\left(W_{l_m} \middle| V(K_{\text{sub}}) \right)}{M} \tag{9.10}$$

Eine zu untersuchende Wortkette W_{l_m} gilt genau dann als „nicht im Vokabular“, wenn mindestens ein Wort aus der Wortkette W_{l_m} nicht im betrachteten Vokabular enthalten ist:

$$\text{oov}\left(W_{l_m} \middle| V(K_{\text{sub}}) \right) = \begin{cases} 1 & \text{, falls mind. ein Wort von } W_{l_m} \text{ nicht in } V(K_{\text{sub}}) \\ 0 & \text{, falls alle Worte von } W_{l_m} \text{ in } V(K_{\text{sub}}) \end{cases} \tag{9.11}$$

9.3 Bestimmung der mittleren OOV-Rate

Im Gegensatz zu vorausgegangenem Kapitel wird nun nicht irgendein beliebiger Subkorpus als Teilmenge des ursprünglichen Korpus‘ K gebildet und anhand des übrigen Testkorpus‘ dessen spezielle OOV-Rate bestimmt, sondern Vokabulargröße und OOV-Rate werden für alle $\binom{L}{M}$ Konstellationen von verschiedenen Subkorpora und Testkorpora bei jeweils gleicher Testkorpusgröße $\left|K_{\text{test}}\right| = M = \text{const.}$ und $1 \leq M \leq L-1$ berechnet und gemittelt.

Für die mittlere Vokabulargröße $\varphi_K(M)$ ergibt sich in Abhängigkeit von der jeweiligen Testkorpusgröße M:

$$\varphi_K(M) = \overline{V(K_{\text{sub}})} = \frac{\sum_{\substack{\text{alle Komb. aus} \\ K_{\text{sub}} \text{ und } K_{\text{test}}}} V(K_{\text{sub}})}{\binom{L}{M}} \tag{9.12}$$

Für die mittlere OOV-Rate $\omega_K(M)$ ergibt sich in Abhängigkeit von der jeweiligen Testkorpusgröße M:

$$\omega_K(M) = \overline{OOV(K_{\text{sub}})} = \frac{\sum_{\substack{\text{alle Komb. aus} \\ K_{\text{sub}} \text{ und } K_{\text{test}}}} OOV(K_{\text{sub}})}{\binom{L}{M}} \tag{9.13}$$

Mit den aus Gl. (9.12) und (9.13) bestimmten Werten können die mittlere Vokabulargröße und mittlere OOV-Rate des um M Wortketten verminderten Korpus K angegeben werden.

Der Rechenaufwand ist jedoch teilweise erheblich, da hierfür $\binom{L}{M}$ Kombinationen mit insgesamt $M \cdot \binom{L}{M}$ Wortketten untersucht werden müssen.[45] Um den Rechenaufwand sinnvoll zu reduzieren, kann auch eine (zufällig gebildete) Stichprobe aus den insgesamt $\binom{L}{M}$ denkbaren Kombinationen gebildet und daraus die mittlere Vokabulargröße $\varphi_{K,S}(M)$ und die mittlere OOV-Rate $\omega_{K,S}(M)$ berechnet werden. Bei genügend großer Stichprobe[46] kann angenommen werden:

$$\varphi_K(M) \approx \varphi_{K,S}(M) \tag{9.14}$$

$$\omega_K(M) \approx \omega_{K,S}(M) \tag{9.15}$$

Ist der verbleibende Subkorpus mit dem gesamten Korpus K identisch (dies entspräche $M = 0$), steht demnach das komplette Vokabular zur Verfügung:

$$\varphi_K(0) = |V(K)| = T \tag{9.16}$$

Besteht der verbleibende Subkorpus aus keiner Wortkette (dies entspräche $M = L$), entartet auch das Vokabular zur leeren Menge $\varnothing$ mit der Vokabulargröße $|\varnothing| = 0$. Da die in einer beliebigen Wortkette enthaltenen Worte nicht in einem leeren Vokabular enthalten sein können, ergibt sich die OOV-Rate zwingend zu 1.

45) Bei $L = 1843$ (Größe des gesammelten Trainingskorpus' innerhalb der Grafikeditor-Domäne) und $M = 20$ wären $\binom{1843}{20} \approx 7,576 \cdot 10^{47}$ Wortketten-Kombinationen und zwanzigmal so viele Wortketten zu untersuchen.

46) Falls das Ergebnis auf drei Nachkommastellen genau sein soll, sind 10^5 zufällige Kombinationen aus K_{sub} und K_{test} sicherlich eine sinnvolle Anzahl.

$$\varphi_K(L) = 0 \tag{9.17}$$

$$\omega_K(L) = 1 \tag{9.18}$$

Diese oben angesprochenen Werte $\varphi_K(M)$ und $\omega_K(M)$ können nun für alle M mit 0 bzw. $1 \leq M \leq L$ bestimmt werden. Es zeigt sich dabei:

- $\varphi_K(M)$ fällt monoton für $0 \leq M \leq L$, d.h. je größer der Testkorpus, desto kleiner der verbleibende Subkorpus, desto kleiner das resultierende Vokabular.
- $\omega_K(M)$ steigt monoton für $1 \leq M \leq L$, d.h. je größer der Testkorpus, desto kleiner der verbleibende Subkorpus, desto größer die resultierende OOV-Rate. Dies erscheint als eine logische Konsequenz, da die Wahrscheinlichkeit zwangsläufig abnehmen muß, daß Worte eines wachsenden Testkorpus‘ in einem schrumpfenden Subkorpus enthalten sind. Damit ist auch die umgekehrte Aussage erklärt, daß die Wahrscheinlichkeit steigt, daß Worte darin nicht enthalten sind.

9.4 Sättigung

Bewirkt eine Korpusvergrößerung keine Vokabularvergrößerung mehr, so ist der bestehende Korpus in einem Zustand der Vollständigkeit bzw. der Sättigung angelangt. In diesem Fall könnte die zeitaufwendige Arbeit des Wortkettensammelns beendet werden. Es werden nun Methoden gesucht, wie ein gegebener Korpus K auf eine eventuell bereits bestehende Sättigung überprüft werden kann. Dazu wird vereinfachend auf eine weitere Wortkettensammlung verzichtet und stattdessen überprüft, wie sich eine Korpusverkleinerung auf das Vokabular auswirkt. Diese Beobachtung kann in erster Näherung Rückschlüsse auf eine voraussichtliche Vokabularänderung bei einer Korpusvergrößerung machen. Dazu wird definiert:

> *Falls eine Korpusverkleinerung um σ beliebige Wortketten keine Vokabularverkleinerung hervorruft, dann gilt dieser Korpus als σ-fach gesättigt.*

Ein Korpus K gelte als einfach gesättigt, wenn

$$\varphi_K(1) = \varphi_K(0) = |V(K)| \quad \text{und damit auch} \quad \omega_K(1) = 0 \tag{9.19}$$

erfüllt ist. Ein Korpus K gelte als σ-fach gesättigt, wenn feststeht:

$$\begin{aligned} \varphi_K(\sigma) = \varphi_K(\sigma-1) = \ldots = \varphi_K(1) = \varphi_K(0) = |V(K)| \qquad &\text{und} \\ \omega_K(\sigma) = \omega_K(\sigma-1) = \ldots = \omega_K(1) = 0 \end{aligned} \tag{9.20}$$

In der Praxis wird in der Regel bereits ein einfach gesättigter Korpus, der unter realen Bedingungen (z.B. durch eine „Wizard of Oz“-Simulation) gewonnen wurde, selbst mit größtem Aufwand nicht zu erreichen sein. Wenn weitere Wortketten gesammelt werden, wird die Häufigkeit des Auftretens neuer, bisher unbekannter Worte zwar abnehmen. Sie wird aber in der Regel nicht auf Null gehen, was bedeutet, daß beim Vergrößern eines Korpus‘ – selbst bei einem extrem großen Korpus – stets mit einem neuen Wort zu rechnen ist.

9.5 Modell mit analytischen Funktionen

Um für einen vorliegenden, begrenzten Korpus K Voraussagen über Vokabulargröße und OOV-Rate treffen zu können, müssen quasi-kontinuierliche, analytische Funktionsterme $\Phi_K(M)$ und $\Omega_K(M)$ in Abhängigkeit von M gefunden werden. Diese sollten zumindest für kleine M eine möglichst gute Approximation der vorher bestimmten Werte $\varphi_K(M)$ und $\omega_K(M)$ sein und ebenfalls für $M \leq 0$ definiert sein:

$$\Phi_K(M) \approx \varphi_K(M) \tag{9.21}$$

$$\Omega_K(M) \approx \omega_K(M) \tag{9.22}$$

Die OOV-Rate des Korpus‘ K ist diejenige Wahrscheinlichkeit, mit der mindestens ein Wort einer zu erwartenden Wortkette innerhalb der betrachteten Domäne nicht im Vokabular $V(K)$ enthalten ist. Diese Größe kann effektiv mit dem Wert $\Omega_K(0)$ angenähert werden. (Falls keine analytische Funktion vorliegt, kann in guter Näherung auch der Wert $\omega_K(1)$ verwendet werden.)

Falls diese Funktionen für negative, ganzzahlige M ebenfalls definiert sind, lassen sich damit die zu erwartende Vokabulargröße und die zu erwartende OOV-Rate abschätzen, wenn man dem Korpus ($-M$) weitere Wortketten zufügt. Dabei wird vorausgesetzt, daß diese hinzuzufügenden Wortketten innerhalb derselben Domäne und unter denselben Bedingungen wie diejenigen im ursprünglichen Korpus K gewonnen werden.

Um eine vorgegebene OOV-Rate ε zu unterschreiten, muß dasjenige M gefunden werden, welches die Bedingung $\Omega_K(M) < \varepsilon$ erfüllt.

9.5.1 Ansatz

Für einen großen, realen Korpus können aufgrund der vorliegenden Werte von $\varphi_K(M)$ und $\omega_K(M)$ folgende Annahmen getroffen werden:

- Die zu bildenden Funktionen $\Phi_K(M)$ und $\Omega_K(M)$ müssen im Bereich $M \leq L$ (auch für $M \leq 0$) definiert sein.
- $\Phi_K(M)$ ist im Definitionsbereich positiv oder Null und streng monoton fallend:

$$\Phi_K(M) \geq 0 \tag{9.23}$$

$$\frac{\partial \Phi_K(M)}{\partial M} < 0 \tag{9.24}$$

- Für $M \to -\infty$ sollte $\Phi_K(M)$ (in „negativer Richtung“ betrachtet) immer weniger zunehmen, was gleichbedeutend ist mit:

$$\lim_{M \to -\infty} \frac{\partial \Phi_K(M)}{\partial M} = 0 \tag{9.25}$$

- $\Omega_K(M)$ ist im Definitionsbereich positiv und streng monoton steigend:

$$\Omega_K(M) > 0 \tag{9.26}$$

$$\frac{\partial \Omega_K(M)}{\partial M} > 0 \tag{9.27}$$

- Für $M \to -\infty$ nähert sich $\Omega_K(M)$ asymptotisch an Null an und sollte (in „negativer Richtung“ betrachtet) immer weniger abnehmen, d.h.:

$$\lim_{M \to -\infty} \Omega_K(M) = 0 \tag{9.28}$$

$$\lim_{M \to -\infty} \frac{\partial \Omega_K(M)}{\partial M} = 0 \tag{9.29}$$

Anhand dieser Voraussetzungen und der experimentell erhaltenen Werte von $\varphi_K(M)$ und $\omega_K(M)$ (siehe dazu Abb. 9.2) werden nun geeignete Funktionsterme gesucht. Die Strukturen folgender Funktionen erfüllen bei geeigneten Werten von a_1, a_2, a_3 und a_4 die getroffenen Voraussetzungen und lassen sich sehr gut und wesentlich besser als andere Funktionsterme auf die vorliegenden Werte der mittleren Vokabulargröße $\varphi_K(M)$ und der mittleren OOV-Rate $\omega_K(M)$ anpassen:

$$\boxed{\Phi_K(M) = a_1 \cdot \sqrt{a_2 - M} + a_3} \tag{9.30}$$

$$\boxed{\Omega_K(M) = \frac{a_4}{\sqrt{a_2 - M}}} \tag{9.31}$$

Dabei müssen $a_1 > 0$, $a_2 > L$, $a_3 > -a_1\sqrt{a_2 - L}$ und $a_4 > 0$ sein, um sämtliche, oben genannte Voraussetzungen zu erfüllen.

Aus der Struktur dieser Funktionen läßt sich der interessante Zusammenhang entnehmen, daß die OOV-Rate proportional zur ersten Ableitung der Vokabulargröße nach M ist:

$$\Omega_K(M) = -\frac{a_4}{a_1} \cdot \frac{\partial \Phi_K(M)}{\partial M} \tag{9.32}$$

Dies erscheint als eine logische Konsequenz, denn je geringer das Wachstum (in „negativer Richtung“) des Vokabulars ist, desto näher muß sich die OOV-Rate auf den Wert 0 zubewegen. Sollte hypothetisch das Vokabular nicht mehr wachsen, sind demnach alle zu erwartenden Worte im Vokabular enthalten und die OOV-Rate nimmt den Wert Null an. Dies wäre der Fall, wenn dem Korpus unendlich viele Wortketten ($M \to -\infty$) hinzugefügt würden.

Die OOV-Rate des Korpus K läßt sich durch den Term

$$\Omega_K(0) = \frac{a_4}{\sqrt{a_2}} \tag{9.33}$$

annähern. Sollte eine bestimmte OOV-Rate ε unterschritten werden, so ergibt sich aus Gl. (9.31) für die hinzuzufügende Anzahl ($-M$) der Wortketten mit $M < 0$:

$$\frac{a_4}{\sqrt{a_2 - M}} < \varepsilon \quad \Rightarrow \quad -M > \left(\frac{a_4}{\varepsilon}\right)^2 - a_2 \tag{9.34}$$

Die resultierende Vokabulargröße $\Phi(\varepsilon)$ ergibt sich in Abhängigkeit von ε zu:

$$\Phi(\varepsilon) = a_1 \cdot \sqrt{a_2 - M} + a_3 > \frac{a_1 \cdot a_4}{\varepsilon} + a_3 \tag{9.35}$$

Bei Gl. (9.35) ist zu beachten, daß die Vokabulargröße $\Phi(\varepsilon)$ nicht von a_2 abhängt.

9.5.2 Beispiel

Im folgenden sei ein Textkorpus K_{wiz} mit $L = 1843$ Wortketten und einer Vokabulargröße von $|V(K_{\text{wiz}})| = 853$ betrachtet. $\varphi_{K\text{wiz}}(M)$ und $\omega_{K\text{wiz}}(M)$ zeigen in Abhängigkeit von M folgenden Verlauf, wobei es sich um diskrete Funktionswerte, welche mit Gl. (9.12) und Gl. (9.13) bestimmt wurden, handelt.

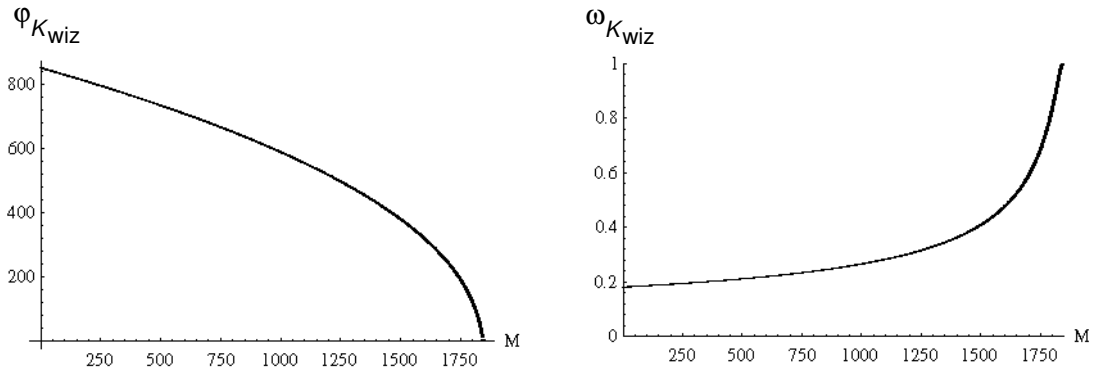


Abb. 9.2: Mittl. Vokabulargröße $\varphi_{K\text{wiz}}(M)$ und mittl. OOV-Rate $\omega_{K\text{wiz}}(M)$ abhängig von M

Zu den ermittelten Werten $\varphi_{K\text{wiz}}(M)$ und $\omega_{K\text{wiz}}(M)$ werden nun mittels Polynomapproximation (Funktion *Fit* des Programms *Mathematica* [Wol91]) Funktionen nach obiger Struktur gefunden. Es ergeben sich folgende analytische Funktionsterme:

$$\Phi_{K_{\text{wiz}}}(M) = 19,5216 \cdot \sqrt{1893 - M} + 6,23525 \tag{9.36}$$

$$\Omega_{K_{\text{wiz}}}(M) = \frac{7,92816}{\sqrt{1893 - M}} \tag{9.37}$$

Die OOV-Rate des Korpus K_{wiz} ergibt sich zu

$$\Omega_{K_{\text{wiz}}}(0) = \frac{7,92816}{\sqrt{1893}} \approx 0,182 \tag{9.38}$$

Dies bedeutet, daß eine zu erwartende Äußerung mit einer Wahrscheinlichkeit von etwa 18,2% mindestens ein dem System unbekanntes Wort enthält.[47] Um eine OOV-Rate von 5% zu unterschreiten, müßten dem Korpus K_{wiz} unter gleichen Bedingungen mindestens

$$-M > \left(\frac{7,92816}{0,05}\right)^2 - 1893 \approx 23250 \tag{9.39}$$

Wortketten hinzugefügt werden. Die Vokabulargröße würde in diesem Fall auf mindestens

$$\Phi_{K_{\text{wiz}}}(\varepsilon = 0,05) > \frac{19,5216 \cdot 7,92816}{0,05} + 6,23525 \approx 3102 \tag{9.40}$$

Worte anwachsen.

9.6 Überprüfung des Modells

Eine einfache und effektive Überprüfung des in den vorausgegangenen Kapiteln aufgestellten Modells ist, aus dem verwendeten Korpus K einen Teilkorpus $\tilde{K}$ zu bilden:

$$\tilde{K} \subset K \tag{9.41}$$

Dieser Korpus $\tilde{K}$ enthält $\tilde{L}$ Wortketten, d.h. die Korpusgröße ist:

$$|\tilde{K}| = \tilde{L} \quad \text{mit} \quad \tilde{L} < L \tag{9.42}$$

Damit werden die entsprechenden Funktionen $\Phi_{\tilde{K}}(M)$ und $\Omega_{\tilde{K}}(M)$ bestimmt. Der Korpus $\tilde{K}$ sollte jedoch groß genug sein, um überhaupt sinnvolle Funktionen zu bilden[48]. Nun kann überprüft werden, ob die mit diesen Funktionen getroffenen Vorhersagen mit den Werten $\Phi_K(M)$ und $\Omega_K(M)$ des vollständigen Korpus‘ K übereinstimmen.

---

47) Dieser Wert ergibt sich unter denselben Bedingungen, die bei der Erstellung des Korpus‘ K_{wiz} herrschten, d.h. bei Spracheingabe durch einen unkundigen Benutzer. Ein kundiger Benutzer paßt sich bewußt oder unbewußt an die Beschränkungen des Systemvokabulars an und vermeidet deshalb den Gebrauch von OOV-Worten, wodurch bei dieser Benutzergruppe die OOV-Rate signifikant sinkt.

48) Als Faustregel kann angenommen werden, daß in diesem Sinne ein Korpus, dessen Korpusgröße deutlich größer als dessen Vokabulargröße ist, d.h. $|\tilde{K}| >> |V(\tilde{K})|$, zur Abschätzung der analytischen Funktionen groß genug erscheint.

$$\Phi_{\tilde{K}}(M - L + \tilde{L}) \stackrel{?}{\approx} \Phi_K(M) \approx \varphi_K(M) \tag{9.43}$$

$$\Omega_{\tilde{K}}(M - L + \tilde{L}) \stackrel{?}{\approx} \Omega_K(M) \approx \omega_K(M) \tag{9.44}$$

Dies wird nun anhand eines konkreten Beispiels überprüft. Der im vorausgegangenen Kapitel verwendete Korpus K_{wiz} wird verkleinert. Jede zweite Wortkette wird extrahiert und damit der Korpus $\tilde{K}_{wiz}$ gebildet. Der neue Korpus enthält $\left|\tilde{K}_{wiz}\right| = \tilde{L} = 922$ Wortketten mit einer resultierenden Vokabulargröße von $\left|V(\tilde{K})\right| = \tilde{T} = 625$ Worten. Es ergeben sich folgende analytische Funktionsterme:

$$\Phi_{\tilde{K}_{wiz}}(M) = 20,8573 \cdot \sqrt{955 - M} - 17,9162 \tag{9.45}$$

$$\Omega_{\tilde{K}_{wiz}}(M) = \frac{7,93053}{\sqrt{955 - M}} \tag{9.46}$$

Nach Gl. (9.43) und Gl. (9.44) sollte für $M = 0$ bzw. $M = 1$ gelten:

$$\begin{aligned} \Phi_{\tilde{K}_{wiz}}(-921) = 885,47 \approx \Phi_{K_{wiz}}(0) = 855,59 \approx \\ \approx \varphi_{K_{wiz}}(0) = 853 \end{aligned} \tag{9.47}$$

$$\begin{aligned} \Omega_{\tilde{K}_{wiz}}(-920) = 0,1831 \approx \Omega_{K_{wiz}}(1) = 0,1822 \approx \\ \approx \omega_{K_{wiz}}(1) = 0,1812 \end{aligned} \tag{9.48}$$

Der relative Fehler bezüglich $\varphi_{K_{wiz}}$ und $\omega_{K_{wiz}}$ beträgt bei der modellierten Vokabulargröße $\Phi_{\tilde{K}_{wiz}}$ 3,8% bzw. bei der modellierten OOV-Rate $\Omega_{\tilde{K}_{wiz}}$ 1,0%.

Werden dem verkleinerten Korpus $\tilde{K}$ 2921 Wortketten hinzugefügt, entspricht dies beim ursprünglichen Korpus K einer Vergrößerung um 2000 Wortketten:

$$\Phi_{\tilde{K}_{wiz}}(-2921) = 1280,61 \approx \Phi_{K_{wiz}}(-2000) = 1224,26 \tag{9.49}$$

$$\Omega_{\tilde{K}_{wiz}}(-2921) = 0,1274 \approx \Omega_{K_{wiz}}(-2000) = 0,1271 \tag{9.50}$$

Der relative Fehler bezüglich $\Phi_{K_{wiz}}$ und $\Omega_{K_{wiz}}$ beträgt bei der modellierten Vokabulargröße $\Phi_{\tilde{K}_{wiz}}$ 4,6% bzw. bei der modellierten OOV-Rate $\Omega_{\tilde{K}_{wiz}}$ 0,2%.

Die geringen relativen Fehler dieser beiden Beispiele zeigen, daß eine Abschätzung der zu erwartenden OOV-Rate und der zu erwartenden Vokabulargröße beim Hinzufügen von einer bestimmten Zahl von Wortketten mit der hier vorgestellten Methode durchaus brauchbare Werte ergibt.

9.7 Schlußbemerkung

Das in diesem Kapitel vorgestellte Verfahren kann und will keine exakte Vorhersage treffen, sondern vielmehr die Möglichkeit eröffnen, die Tendenz und die Größenordnung bezüglich Vokabulargröße und resultierender OOV-Rate abzuschätzen. Es stellt ein relativ einfaches Werkzeug zur Verfügung, mit dem diese Größen ohne weitere zeitaufwendige Korpusvergrößerung ermittelt werden können. Sobald für eine spracherkennende oder sprachverstehende Applikation Trainingsmaterial gesammelt wird, kann mit dieser Abschätzung eine erste Aussage über die Brauchbarkeit, über die Vollständigkeit und damit auch über die Einsetzbarkeit des aus dem Trainingsmaterial gebildeten Vokabulars gemacht werden.

Kapitel 10

Portabilität

10.1 Voraussetzungen

Im Zusammenhang mit dem Entwurf und der Implementierung eines sprachverarbeitenden Systems sollte auf eine einfache Portierbarkeit in eine andere Domäne oder in eine andere Sprache geachtet werden. Grundsätzlich läßt sich im Rahmen der vorliegenden Arbeit der Vorgang des Sprachverstehens (d.h. der Konvertierung eines vorliegenden Sprachsignals in die korrespondierende Intention) als Abfolge von Signalvorverarbeitung, semantischer Decodierung und Intentionsdecodierung, wie in folgender Abbildung gezeigt, auffassen:

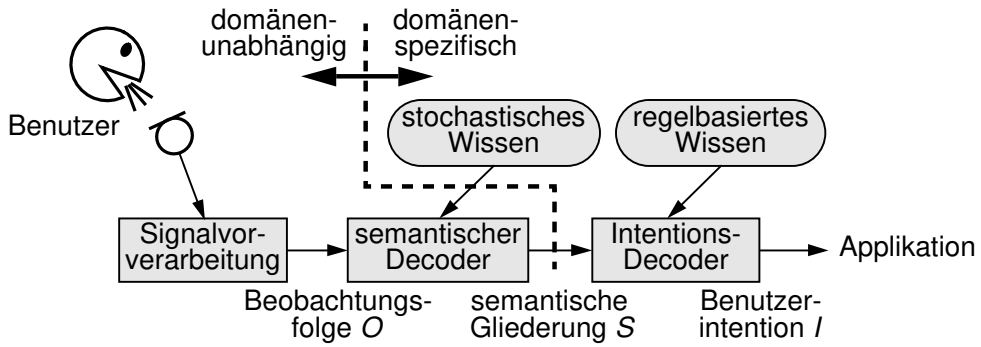


Abb. 10.1: Die sprachverstehende Schnittstelle [Sta97a]

Die Signalvorverarbeitung wandelt ein Sprachsignal lediglich in eine andere Darstellungsform um und ist daher völlig domänen- bzw. sprachunabhängig. Die Algorithmen der semantischen Decodierung konnten ebenfalls domänen- bzw. sprachunabhängig gehalten werden, da sämtliches domänen- bzw. sprachspezifische Wissen zur Laufzeit von externen Dateien eingelesen wird. Lediglich der Intentionsdecoder ist durch seine Nachbarschaft zur eigentlichen Applikation nicht von dieser zu trennen.

Sollte das System innerhalb derselben Domäne oder derselben Applikation auf eine **andere Sprache portiert** werden, betrifft dies bei notwendiger Modifikation des semantischen Inventars die auszugebenen semantischen Gliederungen und deren weitere Verarbeitung innerhalb des Intentionsdecoders. In diesem Fall, wären sämtliche Wissensbasen

des semantischen Decoders und des Intentionsdecoders zu modifizieren. Im Falle von gleichbleibendem semantischem Inventar (siehe dazu Kapitel 8.4) hätte eine Sprachportierung lediglich Änderungen am syntaktischen, am phonetischen und am akustischen Modell des semantischen Decoders zur Folge. Multilinguale Phoneme zum Einsatz innerhalb multilingualer akustischer Modelle sind derzeit im Entwicklungsstadium [Köh96].

Falls bei gleicher Sprache (d.h. gleichbleibendem Phoneminventar) auf eine **andere Domäne portiert** wird, kann lediglich das akustische Modell übernommen werden. Alle anderen stochastischen und regelbasierten Wissensbasen müssen in geeigneter Weise an die neue Domäne angepaßt werden, da die sprachliche Information in einer anderen Domäne vollkommen neuartige Bedeutungen annimmt. Dies hat grundlegende Änderungen am semantischen Modell und an den Wissensbasen des Intentionsdecoders zur Folge. Der neuartige Wortschatz bedingt natürlich darüber hinausgehende Modifikationen am syntaktischen und am phonetischen Modell.

Um eine einfache Portierbarkeit (wie am Beispiel des semantischen Decoders) zu ermöglichen, sollte folgender wichtiger Grundsatz bereits bei der Implementierung eines Moduls beachtet werden:

> *Die Portierung auf eine andere Domäne oder eine andere Sprache sollte ohne Compilierung des Quellcodes möglich sein.*[49)]

10.2 Portierung auf Roboterdomäne

Die innerhalb der NASGRA-Domäne entwickelten Module des semantischen Decoders und des Intentionsdecoders sowie deren erforderliche Wissensbasen sollten zur Bedienung des Serviceroboters ROMAN[50)] zur Ausführung mobiler Handhabungsaufgaben in Innenraumumgebungen [Dax96][Fis96b] portiert werden. Dabei besteht die Aufgabe in der Erweiterung eines bestehenden Mensch-Roboter-Interfaces (MRI)[51)] um einen natürlichsprachlichen Eingabekanal. Der Benutzer soll auch bei dieser Anwendung deutschsprachige, kommandoähnliche Äußerungen ohne Nebensätze und ohne spontansprachliche Effekte verwenden. Die gesamte Funktionalität steht hierbei von vorne herein fest, da sich der abzudeckende Sprach- und Semantikumfang von den verschiedenen an den Roboter gestellten und an anatomischen und planerischen Fähigkeiten orientierten Serviceaufträgen ableitet [Fis96a]:

---

49) Im Klartext bedeutet dies, daß sämtliches domänenspezifische Wissen außerhalb des ausführbaren Programms in speziellen Dateien gespeichert sein sollte. Diese Dateien werden zur Laufzeit anhand ihres Namens in entsprechende Variablen eingelesen. Dafür haben sich innerhalb dieser Arbeit ASCII-Dateien bestens bewährt, da diese im Gegensatz zu reinen Binär-Dateien ohne spezielle Konvertierungsprogramme dargestellt bzw. gelesen werden können.

50) ROMAN bedeutet ROving MANipulator und wurde am Lehrstuhl für Steuerungs- und Regelungstechnik der Technischen Universität München entwickelt.

51) Die bisherigen Eingabearten des MRI sind Kommandos, welche mittels Tastatur in einer speziellen Befehlssyntax eingegeben werden, oder graphische Interaktion, wobei die Objekte in einer virtuellen Welt (AnySIM^{TM}) mittels Mausklick markiert werden.

- Aufnehmen und Abstellen von Objekten, z.B. Geschirr, Bücher oder Werkzeug,
- Bedienen von Einrichtungsgegenständen,
- Transportaufgaben,
- Kontinuierliche Bearbeitung auf Boden, Wand und Mobiliar.

Diese Anwendung stellt eine geeignete und benutzeradäquate Domäne zum Einsatz des Spracheingabekanals dar. Um Gegenstände in der vorher beschriebenden Art zu manipulieren, benutzen wir auch beim Mensch-zu-Mensch-Dialog gesprochene Anweisungen. Natürlichsprachliche Kommandos dieser Art sollten nun von ROMAN verstanden und ausgeführt werden.

Folgende Abbildung zeigt den etwa 1,60 m hohen Serviceroboter ROMAN bei der Ausführung zweier alltäglicher Handhabungsaufgaben.

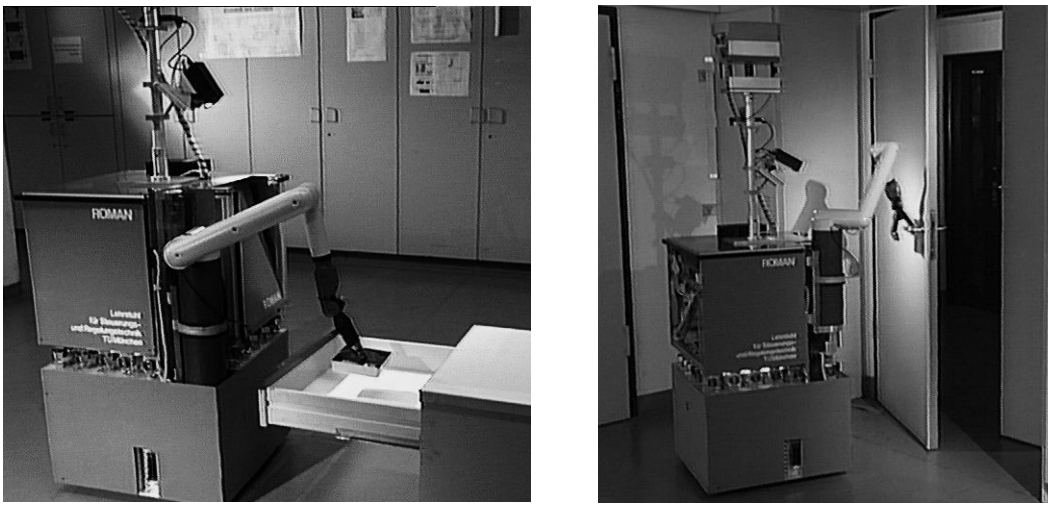


Abb. 10.2: Serviceroboter ROMAN beim Herausnehmen eines Gegenstandes aus einer Schublade (links) und beim Öffnen einer Tür (rechts)

Um eine weitere zeitaufwendige „Wizard of Oz“-Simulation zu umgehen, wurden lediglich denkbare Wortketten zum Bedienen des Serviceroboters gesammelt und damit das semantische und das syntaktische Modell trainiert. Die notwendigen phonetischen Modelle wurden anhand der üblichen lautsprachlichen Transkription manuell erstellt. Der domänenspezifische Intentionsdecoder mußte teilweise modifiziert implementiert werden. Bedingt durch die relativ zur ‚Grafikeditor‘-Domäne geringere semantische Komplexität der möglichen semantischen Gliederungen konnte die korrekte Intentionsdecodierungsrate auf 100% optimiert werden [Hav96].

Im Zusammenhang mit der hier durchgeführten Portierung konnte völlig außer acht gelassen werden, wie diese Befehle vom nachfolgenden Planungsmodul, welches die Roboteraktionen koordiniert und durchführt, weiterverarbeitet werden. (Dieses Planungsmodul gehört nicht zum MRI.) Wichtig für den Intentionsdecoder sind lediglich Syntax und Bedeutung der an das nachfolgende Modul übergebenen Befehle, die als Intention *I* definiert sind. Folgende Abbildung veranschaulicht die Möglichkeiten des MRI.

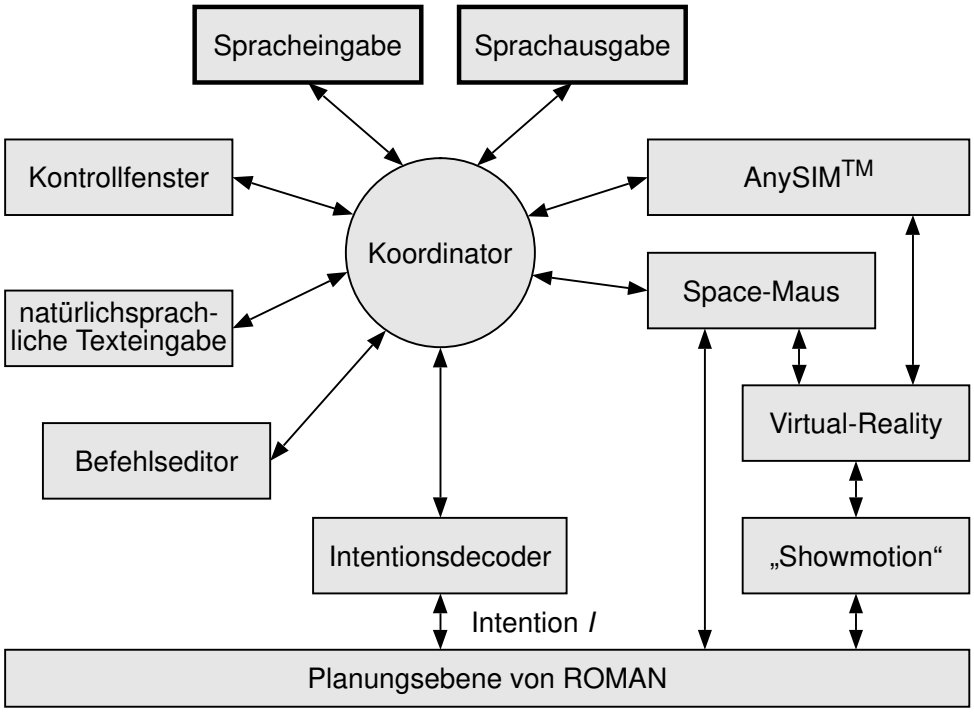


Abb. 10.3: Verwaltungsstruktur und Prozeßkommunikation innerhalb des MRI [Hav96]. Der Befehlsgenerator übernimmt hierbei die prinzipielle Funktion des Intentionsdecoders.

Kapitel 11

Diskussion und Ausblick

Im Rahmen zweier zeitlich parallel ablaufender Arbeiten konnte ein sprachverstehendes System entworfen und für die beiden Domänen „Grafikeditor“ (NASGRA) und „Serviceroboter“ (ROMAN) erfolgreich implementiert werden. Die jeweiligen Demonstrationsapplikationen beweisen anschaulich und interaktiv den komfortablen Einsatz von natürlicher, gesprochener Sprache zur Steuerung eines laufenden Programms.

Der Kernpunkt der vorliegenden Arbeit war der Entwurf der semantischen Gliederung als probabilistisch quantifizierbare Repräsentation des Bedeutungsinhaltes einer sprachlichen Äußerung. Obwohl die semantische Gliederung eine im linguistischen Sinne unpräzise Darstellung bietet, konnte sie sich sowohl innerhalb eines sprachverstehenden Systems zur semantischen Repräsentation als auch innerhalb eines sprachübersetzenden Systems als geeignete Interlingua-Ebene bestens bewähren. Ihr hierarchischer Aufbau kombiniert semantisch-syntaktische Konzepte sehr anschaulich miteinander und erlaubt die Darstellung unendlich vieler Semantikkonstrukte mit einem sehr begrenzten Beschreibungsinventar. Durch eine geeignete Konvertierung dieses Beschreibungsinventars wird eine relativ problemlose Domänenportierung ermöglicht. Da die Entwicklung einer spracherkennenden Applikation und das Erstellen der Wissensbasen eine zeit- und kostenintensive Arbeit darstellt, wurden die Algorithmen der semantischen Decodierung völlig domänen- und applikationsunabhängig implementiert.

Ein wichtiges Kriterium in Richtung möglichst uneingeschränkter sprachlicher Eingaben ist die Verringerung der ‚Out of Vocabulary‘-Rate durch eine Vergrößerung des dem System zur Verfügung stehenden Vokabulars. Eine übliche Methode ist dabei das weitere (und in dieser Weise unendlich lange fortsetzbare) Sammeln von Trainingsmaterial, was jedoch einen zeitintensiven und unkreativen Prozeß darstellt. In diesem Zusammenhang wäre ein selbstlernendes und selbstmodifizierendes System wünschenswert, dem ein Benutzer mittels sprachlicher Korrektur nicht nur neue Worte, sondern auch deren semantische Bedeutung vermitteln könnte. Eine sinnvolle Erweiterung wäre ein adaptives System, das sich auf das Wissen und den Kenntnisstand des Benutzers einstellt. Dabei müßten beim beschriebenen Ansatz nur die zur Verfügung gestellten Wissensbasen zeitlich dynamisch angepaßt werden.

Weiterhin könnte dem in dieser Arbeit vorgestellten System ein geeigneter und robuster Rückweisungsmechanismus hinzugefügt werden, um Erkennungsfehler, die durch ‚Out of Vocabulary'-Worte, Mißachtung der Grammatik oder Domänenverletzung auftreten, zu detektieren. Eine solche Rückweisung müßte an den Benutzer eine geeignete Fehlermeldung ausgeben und könnte eine unerwünschte Aktion innerhalb der nachfolgenden Module rechtzeitig verhindern.

Eine ausschließlich sprachliche Eingabe ist für viele denkbare Anwendungsgebiete – dazu gehört im übrigen auch der in dieser Arbeit beschriebene Grafikeditor – keine vollkommen zufriedenstellende Lösung. Beispielsweise lassen sich innerhalb der ‚Grafikeditor'-Domäne Formen, Farben und Größen sprachlich bestens beschreiben. Bei Positionierungsangaben ist die Sprache jedoch ein denkbar ungeeignetes Eingabemittel. Hier wäre eine Koordinateneingabe direkt mit Berühreingabe[52)] oder indirekt mit der Maus wünschenswert. Auch andere Koordinateneingabeverfahren, die sich derzeit im Entwicklungsstadium befinden, wie Handgesten [Kje96][Mor97] oder Augenstellung (die über ein bildverarbeitendes System oder über Potentialmessung der Haut ausgewertet werden), wären an dieser Stelle sinnvoll. Auch bei der Korrektur von aufgetretenen Erkennungsfehlern erhöhen mehrere bereitgestellte Eingabemodi [Suh96] sowohl die Erkennungsrate als auch die Benutzerakzeptanz. Im natürlichen Mensch-zu-Mensch-Dialog ist ein multimodaler Informationsaustausch selbstverständlich und unerläßlich. Eine Erweiterung eines informationsverarbeitenden Systems auf multimodale Eingabe hätte nicht nur eine kürzere Einarbeitungszeit und leichtere Bedienbarkeit zur Folge, sondern würde beim Benutzer in der Regel eine höhere Akzeptanz erzielen.

Gerade der benutzerseitigen Akzeptanz sollte der Entwickler größte Bedeutung zumessen, zumal der Umgang mit immer komplexer werdenden technischen Systemen auch ungeübten Benutzern Freude und Zufriedenheit bringen soll. Dies kann mit den Methoden des „Usability Engineering" erreicht werden, bei dem auch der (eventuell ungeübte) Benutzer bereits in den Entwicklungsprozeß eingebunden wird. Die Qualität eines technischen Systems sollte im Zuge dieses Vorgehens nicht nur nach einer möglichst großen Funktionalität gemessen werden, sondern es sollten dazu auch subjektive, psychologische und ergonomische Werte Berücksichtigung finden.

Diese Arbeit konnte einen weiteren, erfolgreichen Schritt in Richtung eines benutzeradäquaten, intuitiven Informationsaustausches mit Hilfe der natürlichen, gesprochenen Sprache machen.

52) „Touch-Screen"

Kapitel 12

Anhang

12.1 Glossar

- **Intention**
  ist das, was der Benutzer wirklich will, und bezeichnet diejenige applikationsspezifische Befehlsfolge, die der natürlichsprachlichen Äußerung zugrundeliegt.
- **NASGRA**
  bezeichnet das Projekt „NAtürlichSprachlicher GRAfikeditor“, welches auch im Internet aufzurufen ist: http://www.mmk.e-technik.tu-muenchen.de/~mue/nasgra/
- **‚Out of Vocabulary‘ (OOV)**
  bezeichnet das Auftreten eines Wortes, welches nicht im Systemvokabular enthalten ist, und stellt eine ernste Fehlerquelle für sprachverarbeitende Applikationen dar.
- **ROMAN**
  ist der Serviceroboter des Lehrstuhls für Steuerungs- und Regelungstechnik der Technischen Universität München, welcher nun auch Sprache verstehen kann.
- **Semantische Gliederung**
  ist eine baumförmige, hierarchische Darstellung des Bedeutungsinhalts einer sprachlichen Äußerung, die sich aus mehreren Semunen zusammensetzt.
- **Semun**
  bedeutet „SEMantische UNtereinheit“ oder „SEMantic UNit“, welche einen kleinen Teil der Bedeutung einer sprachlichen Äußerung darstellt.
- **Syntaktisches Modul (SM)**
  ist ein kleines Netzwerk, welches die zu einem Semun gehörende Wortemission regelt.
- **Syntaktisches Netzwerk**
  ist ein aus mehreren SMen hierarchisch aufgebauter probabilistischer Zustands-Übergangs-Automat, der zu einer semantischen Gliederung korrespondiert.
- **Training**
  hat in diesem Zusammenhang nichts mit Sport zu tun und dient zur Abschätzung der Wahrscheinlichkeiten von stochastischen oder probabilistischen Wissensbasen.
- **‚Wizard of Oz‘-Simulation (WOZ)**
  ist eine Versuchsanordnung, bei der die sprachverstehende Funktion einer Applikation von einem menschlichen Zauberer (‚Wizard‘) ohne Wissen des Benutzers simuliert wird. Dies eignet sich bestens zur Gewinnung authentischer Sprachdaten.

12.2 Verwendete Formelzeichen

A_{C_t}	Anzahl der Worte in der Klasse C_t
$A[s_n]$	Syntaktisches Modul (SM) zum Semun s_n
a_n	Pfadwahrscheinlichkeit des Semuns s_n
$a_{n,\mu,\nu}$	Übergangswahrscheinlichkeit vom Zustand μ zum Zustand ν innerhalb des zum Semun s_n gehörenden syntaktischen Moduls
$B[s_n]$	Zustand innerhalb des zum Semun s_n gehörenden SMs, welcher ein bedeutungstragendes Wort emittiert
b_n	Emissionswahrscheinlichkeit des bedeutungstragenden Worts zum Semun s_n
C_t	t-te Klasse in einem Sprachmodell
$C[s_n]$	Zustand innerhalb des zum Semun s_n gehörenden SMs, welcher ein bedeutungsloses Wort emittiert
c_n	Emissionswahrscheinlichkeit des bedeutungslosen Worts zum Semun s_n
D	Schachtelungstiefe einer semantischen Gliederung
d_n	Modulwahrscheinlichkeit des Semuns s_n
e_n	Wertwahrscheinlichkeit des Semuns s_n
f_0	Wurzelwahrscheinlichkeit zum ersten Semun
f_n	Folgewahrscheinlichkeit des Semuns s_n
f_s	Abtastfrequenz
I	Intention (identisch mit maschineninterpretierbarem Code)
I_E	erkannte Intention
K	Textkorpus, bestehend aus mehreren Wortketten
L	Anzahl aller Wortketten eines Textkorpus‘
$L(S)$	Menge aller Linien innerhalb einer semantischen Gliederung S
$\|L(S)\|$	Anzahl aller Linien innerhalb einer semantischen Gliederung S
M	Anzahl der Wortketten, die aus einem Korpus herausgenommen werden
N	Anzahl aller Semune einer semantischen Gliederung
NO	Anzahl aller Merkmalsvektoren einer Beobachtungsfolge
NP	Anzahl aller Phoneme einer Phonemkette
NW	Anzahl aller Worte einer Wortkette
O	Beobachtungsfolge oder Merkmalsvektorenfolge: Lineare Abfolge mehrdimensionaler Merkmalsvektoren, die das Sprachsignal repräsentieren
$\overrightarrow{o_{no}}$	Merkmalsvektor an no-ter Stelle in einer Beobachtungsfolge
Ph	Phonemkette
Ph_{nw}	Teilphonemkette, die zum nw-ten Wort einer Wortkette korrespondiert
ph_{np}	Phonem an np-ter Stelle in einer Phonemkette
q_{bit}	Quantisierungsbreite
$q_x[s_n]$	x-ter Nachfolger des Semuns s_n
S	semantische Gliederung
S_E	erkannte semantische Gliederung
$S(s_n)$	Ast beginnend am Semun s_n
s_n	Semun an n-ter Stelle in einer semantische Gliederung

T	Anzahl aller Worte eines Vokabulars
T_O	zeitliche Dauer einer Beobachtungsfolge O
$t[s_n]$	Typ des Semuns s_n
$t_q(s_n)$	Nachfolgerschar des Semuns s_n
$V(K)$	Vokabular, bestehend aus allen Worten des Textkorpus‘ K
$v[s_n]$	Wert des Semuns s_n
W	Wortkette
W_g	generierte Wortkette (innerhalb der Sprachproduktion)
W_opt	linguistisch nachbearbeitete Wortkette (innerhalb der Sprachproduktion)
$W(s_n)$	Teilwortkette, die zum Ast $S(s_n)$ korrespondiert
W_n	Wortkette an n-ter Stelle im Korpus
w_{nw}	Wort an nw-ter Stelle in einer Wortkette
w_t	Wort an t-ter Stelle im Vokabular
$w^+[s_n]$	bedeutungstragendes Wort zum Semun s_n
$w^-[s_n]$	bedeutungsloses Wort zum Semun s_n
X	Anzahl der Nachfolger eines Semuns
$\gamma[s_n]$	Genus zum Semun s_n (innerhalb der linguistischen Nachbearbeitung)
$\varphi_K(M)$	mittl. Vokabulargröße eines Korpus‘ K abhängig von M
$\varphi_{K,\,S}(M)$	mittl. Vokabulargröße eines Korpus‘ K aus einer Stichprobe abhängig von M
$\Phi_K(M)$	Funktionsterm zur analytischen Beschreibung der mittleren Vokabulargröße
$\kappa[s_n]$	Kasus zum Semun s_n (innerhalb der linguistischen Nachbearbeitung)
$\eta[s_n]$	Numerus zum Semun s_n (innerhalb der linguistischen Nachbearbeitung)
σ	Grad der Sättigung eines Textkorpus‘
$\omega_K(M)$	mittlere OOV-Rate eines Korpus‘ K abhängig von M
$\omega_{K,S}(M)$	mittlere OOV-Rate eines Korpus‘ K aus einer Stichprobe abhängig von M
$\Omega_K(M)$	Funktionsterm zur analytischen Beschreibung der mittleren OOV-Rate

12.3 Trainingskorpora

12.3.1 Domäne Grafikeditor (Auszug von 1843 Wortketten)

```
blauer zylinder ueber der kugel -
den blauen zylinder drehen neunzig_grad im_uhrzeigersinn
- eine gruene pyramide vor die_beiden objekte - auf dem bildschirm
dann - einen gruenen kegel vor die_beiden objekte
- alle objekte in_die rechte obere ecke
objekte_explodieren_lassen und programm beenden
erzeuge zylinder
farbe gruen
erzeuge kegel
vergroessere kugel
verschiebe zylinder links
erzeuge kegel
drehe kegel rechts
beende programm
erzeuge eine grosse schwarze kugel
mache die schwarze kugel weiss
mache die weisse kugel fuenfmal so_gross
bitte - die weisse kugel fuenfmal so_gross
eine mittelgrosse weisse kugel erzeugen bitte
erzeuge eine grosse weisse kugel
bitte die weisse kugel fuenfmal so_gross machen
bitte die weisse kugel ganz an_den unteren_rand schieben
bitte eine kleine rote kugel erzeugen
bitte erzeuge eine grosse weisse kugel
bitte die grosse weisse kugel ganz an_den unteren_rand verschieben
erzeuge bitte eine mittelgrosse weisse kugel
bitte - die kleine weisse kugel dreimal so_gross
bitte die kleine weisse kugel mittig auf die grosse weisse - setzen
bitte - die kleine weisse kugel etwas nach_unten
eine kleine weisse kugel erzeugen bitte
- die kleinste weisse kugel bitte in_den vordergrund
erzeuge einen mittelgrossen roten kegel
setze den kegel auf die mittlere weisse kugel bitte
mache den roten kegel bitte doppelt so_gross
- bitte alle kleinen kugeln in_den vordergrund
bitte mache die untere schwarze kugel doppelt so_gross
bitte mache die grosse schwarze kugel um ein_drittel kleiner
bitte grafik drucken
programm beenden
erzeuge unter der kugel einen roten quader
verschiebe gruene kugel nach_oben und vergroessere sie
erzeuge unter der kugel einen roten quader
vertausche beide objekte und mache sie groesser
erzeuge in_jeder_ecke eine kleine blaue kugel
erzeuge auch in_den_anderen_beiden_ecken eine kleine blaue kugel
erzeuge in_jeder_ecke eine kleine blaue kugel
erzeuge auch in_den_anderen_beiden_ecken eine kleine blaue kugel
loesche die kugel in_der rechten oberen ecke
mache die kugel in_der linken unteren ecke groesser_als die gruene kugel
verschiebe den roten quader ein_kleines bisschen nach_links
erzeuge zwischen der grossen kugel und dem quader einen gelben zylinder
aendere_die_farbe der gruenen kugel
mache die gruene kugel rot
mache die rote kugel heller
mache alle objekte etwas kleiner und blau
vertausche die_beiden zylinder und mache sie groesser
faerbe die_beiden zylinder mit_einem sehr hellen rot
mache den gelben kegel groesser und verschiebe ihn nach_oben
erzeuge ueber dem blauen quader drei gruene kegel mittlerer_groesse
erzeuge einen gruenen kegel ueber dem blauen quader
zeichne in_der mitte eine riesige blaue kugel
```

```
erzeuge rechts_neben der blauen kugel zwei kleine gelbe wuerfel
verschiebe den unteren wuerfel etwas nach_unten
drehe den rechten wuerfel in irgendeine_richtung
setze an_die_stelle der kleinen kugel einen grossen gruenen kegel
erzeuge links unterhalb des linken wuerfels eine kleine gruene kugel
verschiebe die gruene kugel etwas nach_links
erzeuge vor der blauen kugel eine kleine gelbe kugel
kippe den gruenen kegel um fuenfundvierzig_grad nach_rechts
drehe den unteren kegel um hundertachzig_grad
mache alle gelben objekte braun
loesche alle gruenen und braunen objekte
erzeuge links oberhalb der blauen kugel eine weisse mittelgrosse kugel
erzeuge ueber jeder weissen kugel einen braunen mittelgrossen wuerfel
erzeuge ueber jeder weissen kugel einen mittelgrossen braunen wuerfel
kippe den linken wuerfel etwas nach_links
erzeuge unterhalb der blauen kugel einen breiten gruenen quader
verschiebe den gruenen quader etwas nach_unten
okay das_wars
bilde mir ein quadrat
bilde mir einen quader
vergroessere den quader
streiche den quader violett
drehe den quader um neunzig_grad
drehe den quader um fuenfundvierzig_grad
male den quader hell blau
male die kugel weiss
erzeuge eine kugel
erzeuge einen kegel
male den kegel hell blau
positioniere den kegel oberhalb des quaders
verschiebe den kegel um einen_zentimeter nach_unten
positioniere die kugel oberhalb des kegels
mache einen blauen zylinder
mache das letzte kommando rueckgaengig
bilde einen zylinder
male den zylinder weiss an
positioniere die kugel direkt auf_die_spitze des kegels
vergroessere den quader
das letzte kommando bitte nochmal
stelle den zylinder direkt unterhalb des quaders
streiche die kugel hell gruen
erzeuge zwei kugeln
male die zweite kugel in braun
mache das letzte kommando rueckgaengig
streiche die rechte kugel braun
stelle die rechte kugel an_die rechte_ecke des quaders
erzeuge eine braune kugel
stelle die rechte kugel direkt_an_die linke_ecke des quaders
erzeuge einen kegel
verkleinere den rechten kegel
stelle den rechten kegel direkt auf die gruene kugel
erzeuge einen quader
streiche den quader in hell blau
vergroessere den quader
vergroessere den rechten quader
wiederhole das letzte kommando
wiederhole das letzte kommando
wiederhole das letzte kommando nocheinmal
drehe den rechten quader um neunzig_grad
drehe den rechten quader um neunzig_grad
mache den rechten quader laenger
stelle den rechten quader direkt unter den zylinder
beende das programm
nein
den zylinder bitte unter die kugel setzen
dii rosa kugeln zwanzig nach_unten bewegen
setze dii blauen zylinder nebeneinander
(...)
```

12.3.2 Domäne Serviceroboter (Auszug von 285 Wortketten)

```
drehe dich etwas nach_links bitte
bitte ein_bisschen nach_vorn fahren
fahre ein_wenig nach_vorne
noch ein_bisschen nach_rechts drehen bitte
fahre zum mri_tisch
zur sollposition fahren
bewege dich aehm drei_meter rueckwaerts
drehe dich auf neunzig_grad bitte
drehe deine hand um dreissig_grad im_uhrzeigersinn
bewege doch deine hand zur tasse
ziehe_deinen_arm_ein
schau auf den grossen gelben quader
behalte das werkbrett im_auge
auf_den abfalleimer blicken
schau geradeaus
schau noch n_bisschen nach_links
noch ein_stueckchen weiter
mach jetzt bitte weiter
hole aus_dem romanlabor einen orangen schraubenzieher und ein glas
hole zwei rote quader vom tisch_vom_christian
die tuer vom romanlabor bitte oeffnen
hebe den kleinen schwarzen schraubenzieher vom boden auf
transportiere das linke glas zum grosslabor_tisch
gib mir den rechten roten quader
schmeiss ihn auf_deine ladeflaeche
hol mir nochmal drei -
bleib_sofort stehen
stop
breche_alles_ab
mach jetzt nur_noch lokomotion
ab_jetzt nur_noch mobile_manipulation ausfuehren
koenntest_du_bitte zum anrueckpunkt fahren
ein_wenig nach_rechts drehen bitte
fahre noch mehr nach_vorn
zum greifpunkt fahren bitte
du sollst ins grosslabor gehen
geh zum glas
noch weiter drehen
mach den greifer zu
fasse vorsichtig zu
fest zupacken bitte
oeffne deine hand noch ein_stueck
greife mit sollkraft zu
stelle den greifer auf solloeffnung
drehe den effektor nach_links
schiebe deine hand ein_wenig nach_oben
bewege deine hand zum glas
drehe deine hand auf dreisig_grad bitteschoen
fahre_den_arm_aus
noch mehr nach_links schauen bitte
kuck mal weiter runter
schau_dir den klotz an
den bediener ansehen bitte
fixiere die annaeherungsposition
setze_dein_programm_fort
halt_an
mache nur noch manipulation
loesche die_gesamte befehlskette
vergiss den_letzten befehl
fuehre das programm_zwei aus
nimm den linken -
ich will den grossen -
ich haette gerne den roten -
schalte_alles_ab
noch ein_stueck
```

```
weiter
halt
nimm dir den klotz vom werkbrett und bringe ihn auf christians_tisch
stelle das gruene glas auf_den mri_tisch
stelle es aeh auf_den grosslabor_tisch
mach_mal die tuere auf
fahren_sie bitte in christians_zimmer
zum kleinen roten zylinder fahren bitte
wuerden sie_sich bitte nach_rechts drehen
dreh dich noch weiter
dreh dich doch noch mehr nach_rechts
das_reicht
packe fest zu
lass_los
den_letzten befehl vergessen bitte
vergiss den teilauftrag
den serviceauftrag annulieren bitteschoen
fahr mal nach_links
ein_stueckchen nach_vorne -
bewege die plattform weiter nach_links
stelle dich auf_die abrueckposition
gehe noch_weiter nach_hinten
nach_links schwenken
schwenke dich nach_rechts
hand zumachen bitte
lege das teil auf_der transportposition ab
greife den klotz bitte
nehme das glas entgegen
den schraubenzieher vom grosslabor_tisch herholen bitte
lege den zylinder auf_den tisch nieder und mach die romanlabor tuere auf
versuche die grosslabor tuere zu_oeffnen
mach_es nocheinmal
tu_es nochmal
tue den klotz in_deine_hand
abbrechen
alles abbrechen
in christians_zimmer fahren bitte
stelle das teil auf_die transportposition
stelle das teil auf_das werkbrett
brings mir
fahre zur homeposition bitteschoen
fahr noch ein_wenig weiter
bewege aeh die hand noch ein_wenig
noch nach_rechts - bitteschoen
dreh dich doch auf dreisig_grad
auf aeh null_grad drehen
aeh dreh deine hand
aeh schau nach_rechts
auf zwanzig_zentimeter oeffnen bitte
ein_bisschen oeffnen
aeh_mach noch_weiter zu
mache deine hand fest zu
schliesse vorsichtig deine hand
packe mit_deiner hand zu
drehe das gruene glas aeh nach_links
dreh den klotz nach_rechts bitte
drehe den zylinder doch noch mehr
nimm es
bring mir das glas bitteschoen
fahre in die homeposition
fahre in deine homeposition
hole post aus_dem postfach
gehe zum postfach
hole die post
entsorge den abfall
schmeiss den abfall in_den papierkorb
schmeiss den abfall in_den abfalleimer
(...)
```

12.4 Referenzmodelle

Das Referenzmodell bildet die Basis für sämtliche Trainings- und Decodierungsprogramme. Es beinhaltet das gesamte semantische Inventar (d.h. alle Typen und Werte) der relevanten Domäne, kann nur manuell bearbeitet werden und ist zum Einlesen jeglicher Wissensbasis nötig. Wegen ihrer zentralen Bedeutung werden die verwendeten Referenzmodelle für die Domänen „Grafikeditor“ und „Serviceroboter“ vollständig abgedruckt.

12.4.1 Domäne Grafikeditor

```
#*********************************************************************
# REFERENZMODELL enthält alle möglichen Typen und Werte
# Domäne Grafikeditor (Version 3)
# Anzahl aller Typen:  41
# anzahl aller Werte: 263
#*********************************************************************
achse
1
     bezug unbestimmt x xLinks xRechts xz y z zLinks zRechts ;
achseLog2
2
     und ;
anz
1
     1 2 3 4 5 6 7 8 9 10 alle ;
anzAllg
1
     1 1jeweils 2 3 4 5 6 7 8 9 10 einige ;
ausri
1
     unbestimmt kopf senkr waagr ;
bef
1
     gruppieren loeschen vertauschen ;
befAllg
1
     erzeugen ;
befAusgeben
1
     drucken speichern ;
befBeruehren
2
     void ;
befDrehen
3
     void waagrecht ;
befFaerben
3
     void ;
befFaerSkal
3
     void ;
befIrrelevant
1
     beschimpfen void ;
befIrrelevantPause
1
     void ;
befLog2
2
     und ;
befSchieben
3
```

```
     auseinander ausrichten beruehren void zentrieren ;
befSkalieren
3
     faktor0,25 faktor0,5 faktor2 faktor3 faktor4 faktor5 faktor6 mehr
     mehrBreite mehrHoehe weniger wenigerBreite wenigerHoehe void ;
befSpiegeln
2
     void ;
befSteuern
1
     ende initialisieren korrektur wiederholen ;
befUmformen
2
     void ;
befVervielfachen
2
     faktor2 void ;

farbe
1
     beige bunt blau braun dunkel dunkler durchsichtig gelb grau gruen
     gold hell heller oliv orange rosa rot schwarz silber tuerkis violett
     weiss ;
farbeGroesseLog2
2
     und ;
farbeRel
1
     farbigWie ;
farbeLog2
2
     und ;
form
5
     dreieck kegel kreis kugel pyramide quader quadrat unbestimmt
     viereck wuerfel zylinder ;
formAllg
5
     dreieck kegel kreis kugel pyramide quader quadrat unbestimmt
     viereck wuerfel zylinder ;
formBezug
1
     bezug ;
formLog2
2
     ausser und ;
form1
1
     grafik hintergrund lichtquelle ;
groesse
1
     bezug bezugBreite bezugHoehe bezugTiefe maximal minimal normal mehr
     mehrBreite mehrHoehe mehrTiefe viel vielBreite vielHoehe vielTiefe
     wenig wenigBreite wenigHoehe wenigTiefe weniger wenigerBreite
     wenigerHoehe wenigerTiefe ;
groesseRel
1
     breitWie grossBis grossWie groesserAls ;
groesseLog2
2
     ausser und ;
lage
1
     aussen dort hinten links mitte oben rechts unten vorne ;
lageBew
1
     ganzHinten ganzLinks ganzMitte ganzOben ganzRechts ganzUnten
     ganzVorne nachAussen nachHinten nachLinks nachLinksOben
```

```
     nachLinksUnten nachOben nachRechts nachRechtsOben nachRechtsUnten
     nachUnten nachVorne nachZusammen ;
lageRel
1
     anstelle hinter innerhalb linksNeben mitteVon nach neben
     parallel rechtsNeben ueber unter vor zwischen ;
lageLog2
2
     und ;
quantPrz
1
     etwas viel 10 20 25 30 33 50 60 100 110 150 200 300 400 500 600 800 ;
quantRot
1
     etwas 10grad 15grad 17grad 20grad 25grad 30grad 36grad 37grad
     45grad 60grad 90grad 125grad 135grad 160grad 180grad ;
quantTra
1
     bezugBreite bezugHoehe etwas ganz viel 1mm 2mm 3mm 4mm 5mm 6mm 7mm
     8mm 9mm 10mm 15mm 20mm 30mm 40mm 45mm 50mm 70mm 80mm ;
subst
1
     eingabe programm ;

# Ende des Referenzmodells
```

12.4.2 Domäne Serviceroboter

```
#*********************************************************************
# REFERENZMODELL enthält alle möglichen Typen und Werte
# Domäne Serviceroboter (Version 2)
# Anzahl aller Typen:  42
# anzahl aller Werte: 214
#*********************************************************************
#---------- Befehle zur Lokomotion, Maipulation, Kamera ----------#
befBewegenAbs
2
     void ;
befBewegenRel
3
     void ;
befDrehenAbs
2
     void
;
befDrehenRel
3
     void ;
#---------- Kamerabefehle ----------#
befKameraBewegenAbs
1
     void ;
befKameraBewegenFix
1
     void ;
befKameraBewegenRel
2
     void ;
#---------- Manipualtion ----------#
befManipulation
1
     armEinfalten armAusfalten ;
```

```
befEffektorAbstandAbs
2
     void ;
befEffektorOeffnenRel
2
     void ;
befEffektorSchliessenRel
2
     void ;
befEffektorSchliessenKraft
2
     void ;
#---------- Supervision ----------#
bef
1
     abbrechen stop weitermachen ;
befLoeschen
1
     void ;
befUmschalten
1
     void ;
#---------- Tasks ----------#
befBringen
3
     void ;
befObjekt
2
     abstellen ausschalten einschalten fallenLassen oeffnen schliessen
     suchen ;
befObjektBewegenRel
3
     void ;
befObjektDrehenRel
3
     void ;
befObjektGreifen
3
     void ;
befWiederholen
1
     void ;
#---------- Verknüpfung ----------#
befLog2
2
     und ;
objektLog2
2
     oder und ;
#---------- Alle sonstigen Nachfolger ----------#
anz
1
     1 1allg 2 2allg 3 3allg 4 4allg alle ;
befehl
1
     einzelbefehl teilauftrag serviceauftrag ;
farbe
1
     rot gelb gruen blau orange schwarz violett braun grau ;
groesse
1
     gross klein lang kurz breit duenn ;
lage
1
     links rechts ;
irrelevant
1
     void ;
```

```
komponente
1
     effektor kamera plattform ;
modus
1
     normal nurManipulation nurLokomotion nurMobileManipul prog1 prog2
     prog3 prog4 prog5 teleoperation ;
objekt
4
     becher fernbedienung getraenk glas hahn hebel kaffeedose klotz
     muell post quader schachtel schraube ,schraubenzieher schublade
     stift stuhl tasse tuer tuerSchrank unbestimmt werkzeug zylinder ;
objektBezug
1
     bezug ;
posQuelle
1
     abfalleimer abrueck absetz annaeherung bediener bezug boden
     greifpunkt home kueche ladestation nebenzimmer postfach schachtel
     schrank sollposition tischAllg tischMri tischFischer
     tischGrosslabor transportPos werkbrett wohnzimmer zimmerFischer
     zimmerRomanlabor zimmerGrosslabor ;
posZiel
1
     abfalleimer abrueck absetz annaeherung bediener bezug boden ecke
     greifpunkt home kueche ladestation nebenzimmer postfach
     transportPos schachtel schrank sollposition tischAllg tischMri
     tischFischer tischGrosslabor wohnzimmer werkbrett zimmerFischer
     zimmerRomanlabor zimmerGrosslabor ;
quant
1
     mehr normal viel wenig 3mal ;
quantKraft
1
     mehr normal sollkraft viel wenig 10newton 20newton 30newton
     40newton 50newton ;
quantRotAbs
1
     0grad 30grad 90grad 120grad 135grad 150grad 270grad ;
quantRotRel
1
     wenig mehr 9grad 30grad 45grad 60grad 90grad 120grad 180grad
     minus30grad ;
quantTraAbs
1
     solloeffnung wenig 0m 0.05m 0.1m 0.01m 0.02m 0.03m 0.15m 0.2m ;
quantTraRel
1
     wenig normal mehr 0.1m 0.01m 0.02m 0.03m 1m 2m 3m 4m 5m ;
richtung
1
     bezug nachHinten nachLinks nachOben nachRechts nachUnten nachVorne
     nachUnbestimmt ;

# Ende des Referenzmodells
```

12.5 Flexionsmodell

```
#*********************************************************************
# FLEXIONSMODELL - EMISSIONEN
# Wortflexionen zur Generierung von Deutsch
# Domäne Grafikeditor
#*********************************************************************

#---------------------------------
# Existierende Geni:
#---------------------------------
Maskulinum Femininum Neutrum ;

#---------------------------------
# Existierende Numeri:
#---------------------------------
Singular_Allgemein Plural_Allgemein Singular_Bestimmt
Plural_Bestimmt Singular_Unbestimmt ;

#---------------------------------
# Existierende Kasi:
#---------------------------------
Nominativ Genitiv Dativ Akkusativ ;

#---------------------------------
# Jetzt beginnt die Woerterliste:
#---------------------------------
Maskulinum
kreis
{
     -      -es    -      -      # SingAllg
     -e     -e     -en    -e     # PlurAllg
     -      -es    -      -      # SingBest
     -e     -e     -en    -e     # PlurBest
     -      -es    -      -      # SingUnbe
}

Maskulinum
bildschirm
{
     -      -s     -      -      # SingAllg
     -e     -e     -en    -e     # PlurAllg
     -      -s     -      -      # SingBest
     -e     -e     -en    -e     # PlurBest
     -      -s     -      -      # SingUnbe
}

Maskulinum
kegel quader wuerfel zylinder
{
     -      -s     -      -      # SingAllg
     -      -      -n     -      # PlurAllg
     -      -s     -      -      # SingBest
     -      -      -n     -      # PlurBest
     -      -s     -      -      # SingUnbe
}

Femininum
grafik
{
     -      -      -      -      # SingAllg
     -en    -en    -en    -en    # PlurAllg
     -      -      -      -      # SingBest
     -en    -en    -en    -en    # PlurBest
     -      -      -      -      # SingUnbe
}
```

```
Femininum
lichtquelle pyramide
{
     -       -       -       -       # SingAllg
     -n      -n      -n      -n      # PlurAllg
     -       -       -       -       # SingBest
     -n      -n      -n      -n      # PlurBest
     -       -       -       -       # SingUnbe
}

Femininum
kugel
{
     -       -       -       -       # SingAllg
     -n      -n      -n      -n      # PlurAllg
     -       -       -       -       # SingBest
     -n      -n      -n      -n      # PlurBest
     -       -       -       -       # SingUnbe
}

Neutrum
dreieck objekt programm quadrat viereck
{
     -       -s      -       -       # SingAllg
     -e      -e      -en     -e      # PlurAllg
     -       -s      -       -       # SingBest
     -e      -e      -en     -e      # PlurBest
     -       -s      -       -       # SingUnbe
}

Neutrum
kommando
{
     -       -s      -       -       # SingAllg
     -s      -s      -s      -s      # PlurAllg
     -       -s      -       -       # SingBest
     -s      -s      -s      -s      # PlurBest
     -       -s      -       -       # SingUnbe
}

UNDEFINIERT
ihn
{
     ihn     ihn     ihm     ihn     # SingAllg
     sie     sie     ihnen   sie     # PlurAllg
     ihn     ihn     ihm     ihn     # SingBest      Maskulinum
     sie     sie     ihnen   sie     # PlurBest
     ihn     ihn     ihm     ihn     # SingUnbe

     sie     sie     ihr     sie     # SingAllg
     sie     sie     ihnen   sie     # PlurAllg
     sie     sie     ihr     sie     # SingBest      Femininum
     sie     sie     ihnen   sie     # PlurBest
     sie     sie     ihm     sie     # SingUnbe

     es      es      ihm     es      # SingAllg
     sie     sie     ihnen   sie     # PlurAllg
     es      es      ihm     es      # SingBest      Neutrum
     sie     sie     ihnen   sie     # PlurBest
     sie     sie     ihm     sie     # SingUnbe
}

UNDEFINIERT
bunt blau braun breit breiter duenn duenner dunkler flach flacher
gedreht gelb grau gruen gross groesser hell heller klein kleiner kleinst
kopfstehend kurz kuerzer laenger lang liegend mittelgross oliv riesig
rot schwarz stehend tief tiefer unausgefuellt weiss
{
```

```
     -er     -en     -en     -en     # SingAllg
     -e      -er     -en     -e      # PlurAllg
     -e      -en     -en     -en     # SingBest      Maskulinum
     -en     -en     -en     -en     # PlurBest
     -er     -en     -em     -en     # SingUnbe

     -e      -en     -en     -e      # SingAllg
     -e      -er     -en     -e      # PlurAllg
     -e      -en     -en     -e      # SingBest      Femininum
     -en     -en     -en     -en     # PlurBest
     -e      -er     -er     -e      # SingUnbe

     -es     -en     -en     -es     # SingAllg
     -e      -er     -en     -e      # PlurAllg
     -e      -en     -en     -e      # SingBest      Neutrum
     -en     -en     -en     -en     # PlurBest
     -es     -en     -em     -es     # SingUnbe
}

UNDEFINIERT
der
{
     :       :       :       :       # SingAllg
     :       :       :       :       # PlurAllg
     der     des     dem     den     # SingBest      Maskulinum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe

     :       :       :       :       # SingAllg
     :       :       :       :       # PlurAllg
     die     der     der     die     # SingBest      Femininum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe

     :       :       :       :       # SingAllg
     :       :       :       :       # PlurAllg
     das     des     dem     das     # SingBest      Neutrum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe
}

UNDEFINIERT
ein je_ein
{
     -       -es     -em     -en     # SingAllg
     :       :       :       :       # PlurAllg
     :       :       :       :       # SingBest      Maskulinum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe


     -e      -er     -er     -e      # SingAllg
     :       :       :       :       # PlurAllg
     :       :       :       :       # SingBest      Femininum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe

     -       -es     -em     -       # SingAllg
     :       :       :       :       # PlurAllg
     :       :       :       :       # SingBest      Neutrum
     :       :       :       :       # PlurBest
     :       :       :       :       # SingUnbe
}

(...)
```

12.6 Semantisches Modell zur Decodierung von Deutsch

```
#*********************************************************************
# SEMANTISCHES MODELL mit Wurzel-, Wert- und Folgewahrscheinlichkeiten
# Domäne Grafikeditor (Version 3)
# Anzahl aller Typen:              41
# Anzahl aller Werte:             263
# Anzahl aller Nachfolgerscharen: 235
#*********************************************************************

achse                 0  1
     bezug                                0.0194175
     unbestimmt                           0.00970874
     x                                    0.126214
     xLinks                               0.106796
     xRechts                              0.0679612
     xz                                   0.0291262
     y                                    0.15534
     z                                    0.038835
     zLinks                               0.165049
     zRechts                              0.281553
     >LEER                                1
;
achseLog2             0  2
     und                                  1
     >achse>achse                         1
;
anz                   0  1
     1                                    0.88145
     2                                    0.0386978
     3                                    0.000614251
     4                                    0.002457
     alle                                 0.0767813
     >LEER                                1
;
anzAllg               0  1
     1                                    0.894472
     1jeweils                             0.00251256
     2                                    0.0603015
     3                                    0.0100503
     4                                    0.0150754
     5                                    0.00251256
     6                                    0.00251256
     einige                               0.0125628
     >LEER                                1
;
ausri                 0  1
     unbestimmt                           0.184211
     kopf                                 0.0526316
     senkr                                0.315789
     waagr                                0.447368
     >LEER                                0.815789
     >quantRot                            0.184211
;
bef                   0.0423223  1
     gruppieren                           0.010989
     loeschen                             0.956044
     vertauschen                          0.032967
     >form                                0.89011
     >form1                               0.043956
     >formAllg                            0.010989
     >formLog2                            0.0549451
;
befAllg               0.204015  1
     erzeugen                             1
     >form                                0.055
     >formAllg                            0.94
```

```
     >formLog2                                0.005
;
befAusgeben           0.0162778  1
     drucken                                  0.935484
     speichern                                0.0645161
     >form1                                   0.774194
     >LEER                                    0.193548
     >befIrrelevant                           0.0322581
;
befBeruehren          0.00705372  2
     void                                     1
     >form>formAllg                           0.214286
     >form>form                               0.357143
     >formAllg>form                           0.0714286
     >formLog2>formBezug                      0.142857
     >form>formLog2                           0.142857
     >form>formBezug                          0.0714286
;
befDrehen             0.0732501  3
     void                                     0.993827
     waagrecht                                0.00617284
     >form>achse>quantRot                     0.469136
     >form>achse>LEER                         0.0617284
     >form>LEER>quantRot                      0.271605
     >form>LEER>LEER                          0.0555556
     >formBezug>LEER>LEER                     0.00617284
     >formBezug>achse>quantRot                0.0123457
     >LEER>achse>quantRot                     0.0185185
     >LEER>LEER>quantRot                      0.0308642
     >form1>achse>LEER                        0.0123457
     >form1>achse>quantRot                    0.0123457
     >formBezug>LEER>quantRot                 0.0123457
     >LEER>ausri>LEER                         0.00617284
     >form>ausri>LEER                         0.00617284
     >LEER>achse>LEER                         0.00617284
     >form>achseLog2>quantRot                 0.0123457
     >formLog2>achse>quantRot                 0.00617284
;

befFaerben            0.0954965  3
     void                                     1
     >LEER>farbe>LEER                         0.0353535
     >form>farbe>LEER                         0.838384
     >form>LEER>LEER                          0.00505051
     >form>farbe>quantPrz                     0.0252525
     >formBezug>farbe>LEER                    0.020202
     >form>farbeRel>LEER                      0.00505051
     >formLog2>farbe>LEER                     0.00505051
     >LEER>farbe>quantPrz                     0.00505051
     >formAllg>farbe>LEER                     0.010101
     >form1>farbe>LEER                        0.0353535
     >form>farbeLog2>LEER                     0.010101
     >formBezug>LEER>LEER                     0.00505051
;

befFaerSkal           0.00217037  3
     void                                     1
     >form>farbeGroesseLog2>quantPrz          0.25
     >form>farbeGroesseLog2>LEER              0.5
     >formBezug>farbeGroesseLog2>quantPrz     0.25
;
befIrrelevant         0.00379816  1
     beschimpfen                              0.411765
     void                                     0.588235
     >LEER                                    0.764706
     >form1                                   0.0588235

(...)
```

12.7 Syntaktisches Modell zur Generierung von Englisch

```
#*********************************************************************
# SYNTAKTISCHES MODELL - EMISSIONEN
# Generierung von Englisch
# Domäne Serviceroboter (Version 3)
#*********************************************************************
achse
; #keine bed.losen Worte
     bezug              around_its_axis         1 ;
     unbestimmt         around_any_axis         1 ;
     x                  around_the_x_axis       1 ;
     xLinks             frontwards              1 ;
     xRechts            backwards               1 ;
     xz                 left_backwards          1 ;
     y                  around_the_y_axis       1 ;
     z                  around_the_z_axis       1 ;
     zLinks             counter_clockwise       1 ;
     zRechts            clockwise               1 ;
;
achseLog2
; #keine bed.losen Worte
     und #(Wert)        and                     1 ;
;
anz
; #keine bed.losen Worte
     1                  the                     1 ;
     2                  both_the                1 ;
     3                  the_three               1 ;
     4                  the_four                1 ;
     5                  the_five                1 ;
     6                  the_six                 1 ;
     7                  the_seven               1 ;
     8                  the_eight               1 ;
     9                  the_nine                1 ;
     10                 the_ten                 1 ;
     alle               all_the                 1 ;
;
anzAllg
; #keine bed.losen Worte
     1                  a                       1 ;
     1jeweils           respectively_a          1 ;
     2                  two                     1 ;
     3                  three                   1 ;
     4                  four                    1 ;
     5                  five                    1 ;
     6                  six                     1 ;
     7                  seven                   1 ;
     8                  eight                   1 ;
     9                  nine                    1 ;
     10                 ten                     1 ;
     einige             some                    1 ;
;
ausri
; #keine bed.losen Worte
     unbestimmt         rotated                 1 ;
     kopf               head_standing           1 ;
     senkr              standing                1 ;
     waagr              lying                   1 ;
;
bef
; #keine bed.losen Worte
     gruppieren         group                   1 ;
     loeschen           delete                  1 ;
     vertauschen        change                  1 ;

     ;
```

```
befAllg
; #keine bed.losen Worte
     erzeugen           create                  1 ;
;
befAusgeben
; #keine bed.losen Worte
     speichern          store                   1 ;
     drucken            print                   1 ;
;
befBeruehren
                        with                    1
;
     void               connect                 1 ;
;
befDrehen
; #keine bed.losen Worte
     void               rotate                  1 ;
     waagrecht          lie                     1 ;
;
befFaerben
; #keine bed.losen Worte
     void               paint                   1 ;
;
befFaerSkal
; #keine bed.losen Worte
     void               make                    1 ;
;
befIrrelevant
; #keine bed.losen Worte
     beschimpfen        shit                    1 ;
     void               hallo                   1 ;
;
befIrrelevantPause
; #keine bed.losen Worte
     void               ~                       1 ;
;
befLog2
; #keine bed.losen Worte
     und                and                     1 ;
;
befSchieben
; #keine bed.losen Worte
     auseinander        enlarge_the_distance    1 ;
     ausrichten         align                   1 ;
     beruehren          connect                 1 ;
     void               move                    1 ;
     zentrieren         center                  1 ;
;
befSkalieren
; #keine bed.losen Worte
     faktor0,25         quarter                 1 ;
     faktor0,5          half                    1 ;
     faktor2            double                  1 ;
     faktor3            triple                  1 ;
     faktor4            quadruple               1 ;
     faktor5            quintuple               1 ;
     faktor6            multiply_by_six         1 ;
     mehr               enlarge                 1 ;
     mehrBreite         widen                   1 ;
     mehrHoehe          elongate                1 ;
     weniger            reduce                  1 ;
     wenigerBreite      make_thinner            1 ;
     wenigerHoehe       abridge                 1 ;


(...)
```

```
#*********************************************************************
# SYNTAKTISCHES MODELL - ÜBERGÄNGE
# Generierung von Englisch
# Domäne Grafikeditor (Version 3)
#*********************************************************************
# Wahrscheinlichkeiten in Matrixform
#    P(sta->end) P(sta-> C ) P(sta-> B ) P(sta->Aq1) ...
#    P( C ->end) P( C -> C ) P( C -> B ) P( C ->Aq1) ...
#    P( B ->end) P( B -> C ) P( B -> B ) P( B ->Aq1) ...
#    P(Aq1->end) P(Aq1-> C ) P(Aq1-> B ) P(Aq1->Aq1) ...
#    ...     ...     ...     ...     ...
#*********************************************************************
achse                          # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
achseLog2                      # start->Aq1->B->Aq2->ende
     0 0 0 1 0
     0 0 0 0 0
     0 0 0 0 1
     0 0 1 0 0
     1 0 0 0 0
anz                            # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
anzAllg                        # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
ausri                          # start->Aq1->B->ende
     0 0 0 1
     0 0 0 0
     1 0 0 0
     0 0 1 0
bef                            # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
befAllg                        # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
befAusgeben                    # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
befBeruehren                   # start->B->Aq1->C->Aq2->ende
     0 0 1 0 0
     0 0 0 0 1
     0 0 0 1 0
     0 1 0 0 0
     1 0 0 0 0
befDrehen                      # start->B->Aq1->Aq3->Aq2->ende
     0 0 1 0 0 0
     0 0 0 0 1 0
     0 0 0 1 0 0
     0 0 0 0 0 1
     1 0 0 0 0 0
     0 0 0 0 1 0
befFaerben                     # start->B->Aq1->Aq3->Aq2->ende
     0 0 1 0 0 0
```

```
     0 0 0 0 1 0
     0 0 0 1 0 0
     0 0 0 0 0 1
     1 0 0 0 0 0
     0 0 0 0 1 0
befFaerSkal                  # start->B->Aq1->Aq3->Aq2->ende
     0 0 1 0 0 0
     0 0 0 0 1 0
     0 0 0 1 0 0
     0 0 0 0 0 1
     1 0 0 0 0 0
     0 0 0 0 1 0
befIrrelevant                # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
befIrrelevantPause           # start->Aq1->B->ende
     0 0 0 1
     0 0 0 0
     1 0 0 0
     0 0 1 0
befLog2                      # start->Aq1->B->Aq2->ende
     0 0 0 1 0
     0 0 0 0 0
     0 0 0 0 1
     0 0 1 0 0
     1 0 0 0 0
befSchieben                  # start->B->Aq1->Aq3->Aq2->ende
     0 0 1 0 0 0
     0 0 0 0 1 0
     0 0 0 1 0 0
     0 0 0 0 0 1
     1 0 0 0 0 0
     0 0 0 0 1 0
befSkalieren                 # start->B->Aq1->Aq3->Aq2->ende
     0 0 1 0 0 0
     0 0 0 0 1 0
     0 0 0 1 0 0
     0 0 0 0 0 1
     1 0 0 0 0 0
     0 0 0 0 1 0
befSpiegeln                  # start->B->Aq1->Aq2->ende
     0 0 1 0 0
     0 0 0 0 0
     0 0 0 1 0
     0 0 0 0 1
     1 0 0 0 0
befSteuern                   # start->B->Aq1->ende
     0 0 1 0
     0 0 0 0
     0 0 0 1
     1 0 0 0
befUmformen                  # start->B->Aq1->C->Aq2->ende
     0 0 1 0 0
     0 0 0 0 1
     0 0 0 1 0
     0 1 0 0 0
     1 0 0 0 0
befVervielfachen             # start->B->Aq1->Aq2->ende
     0 0 1 0 0
     0 0 0 0 0
     0 0 0 1 0
     0 0 0 0 1
     1 0 0 0 0

(...)
```

12.8 Grammatikregeln

```
#*********************************************************************
# GRAMMATIKREGELN - GENUSBESTIMMUNG
# zur Generierung von Deutsch
# Domäne Grafikeditor
#*********************************************************************
GENUS

ANZAHL
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          ANZAHL
     ;
     VOR_EIGEN          LOGIK
     ;
;

AUSRI
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          AUSRI
     ;
     VOR_EIGEN          LOGIK
     ;
;

FARBE
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          FARBE
     ;
     VOR_EIGEN          LOGIK
     ;
;

FORM
     SELBST                                 $G
                        formBezug           OBJEKT
                        ;
     ;
;

GROESSE
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          GROESSE
     ;
     VOR_EIGEN          LOGIK
     ;
;

LAGE
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          LAGE
     ;
     VOR_EIGEN          LOGIK
     ;
;

LOGIK
     VOR_EIGEN          FORM
     ;
     VOR_EIGEN          LOGIK
     ;
```

Literatur

Verwendete Abkürzungen und Bezeichnungen für regelmäßig stattfindende Tagungen:

AMS:	Fachgespräch „Autonome mobile Systeme“
DAGA:	Gemeinschaftstagung der Deutschen Arbeitsgemeinschaft für Akustik bzw. Deutsche Jahrestagung für Akustik („Fortschritte der Akustik“)
Eurospeech:	European Conference on Speech Communication and Technology
ICASSP:	International Conference on Acoustics, Speech, and Signal Processing
ICSLP:	International Conference on Spoken Language Processing
ISSD:	International Symposium on Spoken Dialogue
KI:	Deutsche Jahrestagung für Künstliche Intelligenz, Advances in Artificial Intelligence
KONVENS:	Konferenz für Verarbeitung natürlicher Sprache

[Aho88] A.V. Aho, R. Sethi, J.D. Ullmann: *Compilerbau – Teil 1*, Addison-Wesley, Bonn, 1988

[Aue96] M. Auerswald, T. Bub, H. Kirchmann, J. Kroner, J. Schwinn: *Verbmobil Integrations- und Testumgebung, Benutzerhandbuch*, Verbmobil-Report 178, DFKI Kaiserslautern, 1996

[Bau95] J.G. Bauer, H. Stahl, J. Müller: *A One-pass Search Algorithm for Understanding Natural Spoken Time Utterances by Stochastic Models*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 567-570

[Beh95] M. Beham: *Merkmalsextraktion und Regelgewinnung für die automatische Spracherkennung*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1995

[Bia93] P. Bianchini, P. Ferragina, M. Notturno Granieri, L. Tarricone: *New Techniques for Speech Understanding*, Tagungsband ICASSP 1993 (Minneapolis, USA), S. II/127-II/130

[Bli92] J. Blieberger, G.H. Schildt, U. Schmid, S. Stöckler: *Informatik*, Springer-Verlag, Wien, 2.Auflage, 1992

[Blo93] M. Blomberg et al.: *An Experimental Dialogue System: Waxholm*, Tagungsband Eurospeech 1993 (Berlin, Deutschland), S. 1867-1870

[Bon95] A. Bonafonte, J.B. Mariño, E. Lleida: *Semantic Decoding of Speech in Constrained Domains*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 559-562

[Bon96] A. Bonafonte, J.B. Mariño, A. Nogueiras: *SETHOS: The UPC Speech Understanding System*, Tagungsband ICSLP 1996 (Philadelphia, USA), S. 2151-2154

[Bro95] M. Brockhaus, A. Ertl: *Übersetzerbau*, Vorlesungsskript, Institut für Computersprachen, Technische Universität Wien, 1995

[Bru82] H.E. Bruderer: *Maschinelle und maschinenunterstützte Sprachübersetzung*, in H.E. Bruderer (Hrsg.): „Automatische Sprachübersetzung“, Wissenschaftliche Buchgesellschaft, Darmstadt, 1982, S. 357-371

[Bus93] S. Busemann, H.J. Novak: *Generierung natürlicher Sprache*, in G. Görz (Hrsg.): „Einführung in die künstliche Intelligenz“, Addison-Wesley Publishing, Bonn, 1993, S. 499-558

[Cic93] A. Cichocki, R. Unbehauen: *Neural Networks for Optimization and Signal Processing*, Teubner-Verlag, Stuttgart, 1993

[Cha95] L. Chase, R. Rosenfeld, A. Hauptmann, M. Ravishankar, et al.: *Improvements in Language, Lexical, and Phonetic Modeling in Spinx-II*, Tagungsband „ARPA Spoken Language Systems Technology Workshop“ (Austin, USA), 1995, S. 60-65

[Dax96] W. Daxwanger, E. Ettelt, C. Fischer, F. Freyberger, U. Hanebeck, G. Schmid: *ROMAN: Ein mobiler Serviceroboter als persönlicher Assistent in belebten Innenräumen*, in G. Schmid, F. Freyberger (Hrsg.): Tagungsband AMS 1996 (München, Deutschland), Springer-Verlag, Reihe „Informatik aktuell“, 1996, S. 314-333

[Dro62] G. Drosdowski, P. Grebe, R. Köster, W. Mentrup, W. Müller: *Duden Aussprachewörterbuch*, Reihe „Der Große Duden“, Band 6, Dudenverlag (Bibliographisches Institut), Mannheim, 1962

[Dro90] G. Drosdowski, W. Müller, W. Scholze-Stubenrecht, M. Wermke: *Duden Fremdwörterbuch*, Bibliographisches Institut & FA Brockhaus Verlag, Mannheim, 1990

[Dyb93] H. Dybkjaer, N.O. Bernsen, L. Dybkjaer: *Wizard-of-Oz and the Trade-off between Naturalness and Recogniser Constraints*, Tagungsband Eurospeech 1993 (Berlin, Deutschland), S. 947-950

[Ebe95] M. Ebersberger: *Entwurf und Implementierung eines Compiler-Interpreter-Systems für den natürlichsprachlichen Mensch-Maschine-Dialog*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1995

[Ebe96] M. Ebersberger, J. Müller, H. Stahl: *A Compiler-Interpreter-System for Decoding the User's Intention Within a Speech Understanding Application*, Tagungsband KI 1996 (Dresden, Deutschland), S. 61-65

[Fet95] P. Fetter, F. Class, U. Haiber, A. Kaltenmeier, U. Kilian, P. Regel-Brietzmann: *Detection of Unknown Words in Spontaneous Speech*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 1637-1640

[Fis96a] C. Fischer, P. Havel, G. Schmidt, J. Müller, H. Stahl, M. Lang: *Kommandierung eines Serviceroboters mit natürlicher, gesprochener Sprache*, in G. Schmid, F. Freyberger (Hrsg.): Tagungsband AMS 1996 (München, Deutschland), Springer-Verlag, Reihe „Informatik aktuell", 1996, S. 248-261

[Fis96b] C. Fischer, M. Buss, G. Schmidt: *Hierarchical Supervisory Control of Service Robot Using HuMan-Robot-Interface*, Tagungsband IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Osaka, Japan), zur Veröffentlichung eingereicht

[Gau94] J. Gauvin, L. Lamel, M. Adda-Decker: *Developments in Continuous Speech Dictation Using the ARPA WSJ Task*, Tagungsband „ARPA Spoken Language Systems Technology Workshop", 1994

[Gol90] L. Goldschlager, A. Lister: *Informatik – Eine moderne Einführung*, Hanser und Prentice-Hall, Wien und London, 3. Auflage, 1990

[Gör88] G. Görz: *Strukturanalyse natürlicher Sprache*, Addison-Wesley Publishing, Bonn, 1988

[Gra92] T. Gramß, H. Behme, H.W. Strube: *Neuronale Netzwerke zur Worterkennung und -reproduktion*, in: H. Mangold (Hrsg.): „Sprachliche Mensch-Maschine-Kommunikation", Oldenbourg-Verlag, München, 1992

[Gra95] R. Grau: *Entwicklung und Implementierung eines Sprachdialogsystems zur Präsentation des Lehrstuhls*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1995

[Hab80] C.U. Habel, C.R. Rollinger, A. Schmidt, H.J. Schneider: *A Logic-Oriented Approach to Automatic Text Understanding*, in L. Bolc (Hrsg.): „Natural Language Based Computer System", Carl Hanser, München, 1980, S. 57-118

[Hae92] R. Haeb-Umbach, H. Ney: *Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition*, Tagungsband ICASSP 1992 (San Francisco, USA), S. I/13-I/16

[Har93] O. Hartmann: *Dialoggestaltung für automatische Telefon-Informations-Dienste am Beispiel eines Auskunftssystems*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1996

[Hau93] H. Haugeneder, H. Trost: *Beschreibungsformalismen für sprachliches Wissen*, in G. Görz (Hrsg.): „Einführung in die künstliche Intelligenz“, Addison-Wesley Publishing, Bonn, 1993, S. 372-424

[Hav96] P. Havel: *Mensch-Roboter-Interface zur semiautonomen Unterstützung des Serviceroboters ROMAN*, Diplomarbeit, Lehrstuhl für Steuerungs- und Regelungstechnik, Technische Universität München, 1996

[Her93] M. Herweg, W. Hoeppner, H. Horacek, J. Kreyß, H.J. Novak: *Die Bedeutung kognitionswissenschaftlicher Erkenntnisse für die automatische Sprachgenerierung*, in O. Herzog, T. Christaller, D. Schütt (Hrsg.): „Grundlagen und Anwendungen der Künstlichen Intelligenz – 17. Fachtagung für Künstliche Intelligenz“, Springer-Verlag, Reihe „Informatik aktuell“, 1993, S. 198-205

[Hög90] H. Höge: *SPICOS II - a Speech Understanding Dialogue System*, Tagungsband ICSLP 1990 (Kobe, Japan), S. 1313-1316

[Hög93] H. Höge: *Statistische Modelle für die Spracherkennung*, Tagungsband DAGA 1993 (Frankfurt am Main, Deutschland), S. 11-30

[Hua90] X.D. Huang, Y. Ariki, M.A. Jack: *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, Edinburgh, 1990

[Kai95a] J. Kaiser: *Training of the Semantic Models and Extension on Slovene Word Chains for a Speech-Understanding Graphic Editor*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1995

[Kai95b] J. Kaiser, J. Müller, H. Stahl: *Training of the Semantic Model for a Speech Understanding Graphic Editor*, Tagungsband Slovendka sekcija IEEE: Zbornik cetrte Elektrotehniske in racunalniske konference ERK 1995 (Portoroz, Slowenien), Band B, S. 285-288

[Kat94] I. Katunobu et al.: *Collecting and Analyzing Nonverbal Elements for Maintenance of Dialog Using a Wizard of Oz Simulation*, Tagungsband ICSLP 1994 (Yokohama, Japan), S. 907-910

[Kin92] W. Kinnebrock: *Neuronale Netze*, Oldenbourg-Verlag, München, 1992

[Kje96] R. Kjeldsen, J. Kender: *Toward the Use of Gesture in Traditional User Interfaces*, Tagungsband ‚International Conference on Automatic Face and Gesture Recognition‘ (Killington, USA), 1996, S. 151-156

[Klo95] R. Klotz: *Richtlinien zur Sprachdialoggestaltung im Sinne einer optimierten Mensch-Maschine-Kommunikation*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1995

[Köh96] J. Köhler: *Multi-Lingual Phoneme Recognition Exploiting Acoustic-Phonetic Similarities of Sounds*, Tagungsband ICSLP 1996 (Philadelphia, USA), S. 2195-2198

[Kor96] B. Kormann: *Optimierung des Benutzerdialogs für einen sprachverstehenden Grafikeditor*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1996

[Kow79] R.A. Kowalski: *Logic for Problem Solving*, Elsevier North Holland, New York, 1979

[Lam95a] L.F. Lamel, S.K. Bennacef, H. Bonneau-Maynard, S. Rosset, J.L. Gauvain: *Recent Developments in Spoken Language Systems for Information Retrieval*, Tagungsband „ESCA Workshop on Spoken Dialogue Systems (Visgø, Dänemark), 1995, S. 17-20

[Lam95b] L.F. Lamel, S. Rosset, S. Bennacef, H. Bonneau-Maynard, L. Devillers, J.L. Gauvain: *Development of Spoken Language Corpora for Travel Information*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 1961-1964

[Lan84] M. Lang: *Computer lernen hören und sehen*, Siemens-Zeitschrift, Band 58 (1984), Heft 3, S. 25-26

[Lan86] M. Lang: *Spracheingabe zur Steuerung automatisierter Systeme. Probleme der Sprachverarbeitung, Stand der Technik und wirtschftliche Aspekte*, VDI-Berichte Nr. 587 (1987), VDI-Verlag, Düsseldorf

[Lan91] M. Lang: *Kommunikation zwischen Mensch und Maschine*, Fachzeitschrift „Mikroelektronik“, Band 5 (1991), Heft 6, S. 212-219

[Lan94a] M. Lang, H. Stahl: *Spracherkennung für einen ergonomischen Mensch-Maschine-Dialog*, Fachzeitschrift „Mikroelektronik“, Band 8 (1994), Heft 2, S. 78-82

[Lan94b] M. Lang: *Mensch-Maschine-Kommunikation 1*, Vorlesungsskript, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, Ausgabe 6/94 (1994)

[Lan94c] M. Lang: *Toward User Adequate Human-Computer-Interaction*, in B. Horvat, Z. Kacic (Hrsg.): Tagungsband „Modern Modes of Man-Machine-Communication“ (Maribor, Slowenien), 1994, S. 1.1-1.9

[Lan94d] M. Lang: *Aspects of Human-Machine-Communication*, in H. Niemann, R. de Mori, G. Hanrieder (Hrsg.): Tagungsband „Progress and Prospects of Speech Research and Technology“ (München, Deutschland), infix, 1994, S. 1-8

[Lan95] M. Lang: *Mensch-Maschine-Kommunikation 2*, Vorlesungsskript, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, Ausgabe 4/95 (1995)

[Lev95] E. Levin, R. Pieraccini: *Concept-Based Spontaneous Speech Understanding System*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 555-558

[Lin94] A. Linke, M. Nussbaumer, P.R. Portmann: *Studienbuch Linguistik*, Max Niemeyer Verlag, Tübingen, 2. Auflage, 1994

[Loh82] S.C. Loh, L. Kong: *Automatische Übersetzung chinesischer wissenschaftlicher Zeitschriften*, in H.E. Bruderer (Hrsg.): „Automatische Sprachübersetzung“, Wissenschaftliche Buchgesellschaft, Darmstadt, 1982, S. 344-356

[Lov78] D.W. Loveland: *Automated Theorem Proving*, North Holland, Amsterdam, 1978

[Maa82] H.D. Maas: *Das Saarbrücker automatische Übersetzungssystem SUSY*, in H.E. Bruderer (Hrsg.): „Automatische Sprachübersetzung“, Wissenschaftliche Buchgesellschaft, Darmstadt, 1982, S. 337-343

[Mar66] H. Marko: *Die Theorie der bidirektionalen Kommunikation und ihre Anwendung auf die Informationsübermittlung zwischen Menschen (subjektive Information)*, Kybernetik 3 (1966), S. 128-138

[Mar94] A. Marx: *Generieren und Optimieren von Sprachmodellen sowie regelbasiertes Textverstehen am Beispiel eines sprachgesteuerten Grafik-Editors*, Diplomarbeit, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 1994

[Mor97] P. Morguet, M. Lang: *Feature Extraction Methods for Consistent Spatio-Temporal Image Sequence Classification Using Hidden Markov Models*, Tagungsband ICASSP 1997 (München, Deutschland), S. 2893-2896

[Mül90] B. Müller, J. Reinhardt: *Neural Networks: An Introduction*, Springer-Verlag, Berlin, 1990

[Mül86] D. Müller-Böling, M. Müller: *Akzeptanzfaktoren in der Bürokommunikation*, Oldenbourg Verlag, München, 1986

[Mül94] J. Müller, H. Stahl: *Ein Ansatz zum Verstehen natürlicher, gesprochener Sprache durch hierarchisch strukturierte Hidden-Markov-Modelle*, Tagungsband KONVENS 1994 (Wien, Österreich), S. 260-269

[Mül95a] J. Müller, H. Stahl: *Die semantische Gliederung als adäquate semantische Repräsentationsebene für einen sprachverstehenden ‚Grafikeditor‘*, in L. Hitzenberger (Hrsg.): Angewandte Computerlinguistik, Reihe „Sprache und Computer“ (Band15), Georg Olms Verlag, Hildesheim, 1995, S. 211-225

[Mül95b] J. Müller, H. Stahl: *Stochastic Modelling of Syntax and Semantics*, Tagungsband KI 1995 Activities: Workshops, Posters, Demos (Bielefeld, Deutschland), S. 229-230

[Mül95c] J. Müller, H. Stahl: *Collecting and Analyzing Spoken Utterances for a Speech Controlled Application*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 1437-1440

[Mül96a] J. Müller, H. Stahl, M. Lang: *Automatic Speech Translation Based on the Semantic Structure*, Tagungsband ICSLP 1996 (Philadelphia, USA), S. 658-661

[Mül96b] J. Müller, H. Stahl, M. Lang: *Predicting the Out-of-Vocabulary Rate and the Required Vocabulary Size for Speech Processing Applications*, Tagungsband ICSLP 1996 (Philadelphia, USA), S. 1922-1925

[Ney84] H. Ney: *The Use of a One-stage Dynamic Programming Algorithm for Connected Word Recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Band ASSP-32 (1984), Nr. 2, S. 263-271

[Oer93] M. Oerder, H. Ney: *Word Graphs: An Efficient Interface Between Continuous Speech Recognition and Language Understanding*, Tagungsband ICASSP 1993 (Minneapolis, USA), S. II/119-II/122

[Pel75] H. Pelz: *Linguistik für Anfänger*, Hoffmann und Campe, Hamburg, 1975

[Pet90] D. Petkovic: *SQL – die Datenbanksprache*, McGraw-Hill, 1990

[Pie91a] R. Pieraccini, E. Levin, C.H. Lee: *Stochastic Representation of Conceptual Structure in the ATIS Task*, Tagungsband „4th DARPA Workshop on Speech and Natural Language“ (Asilomar, USA), 1991

[Pie91b] R. Pieraccini, E. Levin: *Stochastic Representation of Semantic Structure for Speech Understanding*, Tagungsband Eurospeech 1991 (Genua, Italien), S. 383-386

[Pie92a] R. Pieraccini, E. Tzoukermann, et al.: *A Speech Understanding System Based on Statistical Representation of Semantics*, Tagungsband ICASSP 1992 (San Francisco, USA), S. I.193-I.196

[Pie92b] R. Pieraccini, E. Levin: *Stochastic Representation of Semantic Structure for Speech Understanding*, Speech Communication, Band 11 (1992), Heft 2-3, S. 283-288

[Pie92c] R. Pieraccini, Z. Gorelov, E. Levin, E. Tzoukermann: *Automatic Learning in Spoken Language Understanding*, Tagungsband ICSLP 1992 (Banff, Kanada), S. 405-408

[Pie93] R. Pieraccini, E. Levin: *Learning How To Understand Language*, Tagungsband Eurospeech 1993 (Berlin, Deutschland), S. 1407-1412

[Pie95] R. Pieraccini, E. Levin: *A Spontaneous-Speech Understanding System for Database Query Applications*, Tagungsband „ESCA Workshop on Spoken Dialogue Systems“ (Visgø, Dänemark), 1995, S. 85-88

[Pin93] M. Pinkal: *Semantik*, in G. Görz (Hrsg.): „Einführung in die künstliche Intelligenz“, Addison-Wesley Publishing, Bonn, 1993, S. 425-498

[Pla95] B. Plannerer: *Erkennung fließender Sprache mit integrierten Suchmethoden*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1995

[Pri90] P.J. Price: *Evaluation of Spoken Language Systems: The ATIS Domain*, Tagungsband „3rd DARPA Workshop on Speech and Natural Language“ (Hidden Valley, USA), 1990, S. 91-95

[Pri92] N. Prieto, E. Vidal: *Learning Language Models through the ECGI Method*, Speech Communication, Band 11 (1992), Heft 2-3, S. 299-309

[Pri94] N. Prieto, E. Sanchis, L. Palmero: *Continuous Speech Understanding Based on Automatic Learning of Acoustic and Semantic Models*, Tagungsband ICSLP 1994 (Yokohama, Japan), S. 2175-2178

[Rab89] L.R. Rabiner: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proc. IEEE, Band 77 (1989), Heft 2, S. 257-286

[Rei81] R. Reichwald: *Zur Notwendigkeit der der Akzeptanzforschung bei der Entwicklung neuer Systeme der Bürotechnik*, in „Akzeptanz neuer Bürotechnologien“, AKZENTE Studiengemeinschaft, Düsseldorf, 1981

[Rei96] W. Reichl: *Diskriminative Lernverfahren für die automatische Spracherkennung*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1996

[Ros95] R. Rosenfeld: *Optimizing Lexical and N-gram Coverage via Judicious Use of Linguistic Data*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 1763-1766

[Rul89] H. Rulot, N. Prieto, E. Vidal: *Learning Accurate Finite-State Structural Models of Words through the ECGI Algorithms*, Tagungsband ICASSP 1989 (Glasgow, Schottland), S. 643-646

[Rus94] G. Ruske: *Automatische Spracherkennung*, Oldenbourg-Verlag, München Wien, 1994

[Sag90] G. Sagerer: *Automatisches Verstehen gesprochener Sprache*, Habilitation, Reihe Informatik, Band 74, BI-Wissenschaftsverlag, Mannheim, 1990

[Sch91] M. Schwanke: *Maschinelle Übersetzung*, Springer-Verlag, Berlin, 1991

[Sha49] C.E. Shannon, W. Weaver: *The Mathematical Theory of Communication*, The University of Illinois Press, Urbana, 1949

[Sta94] H. Stahl, J. Müller: *An Approach to Natural Speech Understanding Based on Stochastic Models in a Hierarchical Structure*, in B. Horvat, Z. Kacic (Hrsg.): Tagungsband „Modern Modes of Man-Machine-Communication“ (Maribor, Slowenien), 1994, S. 16.1-16.9

[Sta95] H. Stahl, J. Müller: *A Stochastic Grammar for Isolated Representation of Syntactic and Semantic Knowledge*, Tagungsband Eurospeech 1995 (Madrid, Spanien), S. 551-554

[Sta96] H. Stahl, J. Müller, M. Lang: *An Efficient top-down Parsing Algorithm for Understanding Speech by Using Stochastic Syntactic and Semantic Models*, Tagungsband ICASSP 1996 (Atlanta, USA), S. 397-400

[Sta97a] H. Stahl, J. Müller, M. Lang: *Controlling Limited-Domain Applications by Probabilistic Semantic Decoding of Natural Speech*, Tagungsband ICASSP 1997 (München, Deutschland), S. 1163-1166

[Sta97b] H. Stahl: *Konsistente Integration stochastischer Wissensquellen zur semantischen Decodierung gesprochener Äußerungen*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1997

[Suh96] B. Suhm, B. Myers, A. Waibel: *Interactive Recovery from Speech Recognition Errors in Speech User Interfaces*, Tagungsband ICSLP 1996 (Philadelphia, USA), S. 865-868

[Tes65] L. Tesnière: *Éléments de syntaxe structurale*, Klincksieck, Paris, 1965

[Tzo92] E. Tzoukermann, R. Pieraccini, Z. Gorelov, *Natural Language Processing in the CHRONUS System*, Tagungsband ICSLP 1992 (Banff, Kanada), S. 1415-1418

[Vit73] A.J. Viterbi: *Error Bounds for Convolutional Codes and an Asymptotical Optimal Decoding Algorithm*, IEEE Trans. Information Theory, Band 61 (1973), S. 268-278

[Wah93] W. Wahlster: *Verbmobil – Translation of Face-to-Face Dialogs*, Tagungsband Eurospeech 1993 (Berlin, Deutschland), Zusatzband „Opening and Plenary Sessions“, S. 29-38

[Wel95] B.B. Welch: *Practical Programming in Tcl and Tk*, Prentice Hall, New Jersey, 1995

[Win87] P.H. Winston: *Künstliche Intelligenz*, Addison-Wesley Publishing, Bonn, 1987

[Wol91] S. Wolfram: *Mathematica - a System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, 1991

[Wol96] F. Wolfertstetter: *Verallgemeinerte stochastische Modellierung für die automatische Spracherkennung*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1996

[Woo95] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, S.J. Young: *The Development of the 1994 HTK Large Vocabulary Speech Recognition System*, Tagungsband „ARPA Spoken Language Technology Workshop“ (Austin, USA), 1995, S. 104-109

[Wot89] K. Wothke et al.: *The SPRING Speech Recognition System for German*, Tagungsband Eurospeech 1989 (Paris, Frankreich), Band 2, S. 9-12

[Yng92] J. Yngvesson, I. Wallin: *User‘s Guide to SIPP - a 3D Rendering Library*, Version 3.0, 1992

[Zol82] E. Zoltan, G.D. Weeks, W.R. Ford: *Natural-language Communication with Computers: A Comparison of Voice and Keyboard Inputs*, in G. Johannsen & J.E. Rijnsdorp (Hrsg.): Analysis, Design and Evaluation of Man-Machine-Systems, Verein Deutscher Ingenieure, Düsseldorf, 1982, S. 287-292

[Zol91] E. Zoltan-Ford: *How to Get People to Say and Type what Computers Can Understand*, International Journal of Man-Machine-Studies, Band 34 (1991), Heft 4, S. 527-547

[Zün91] K. Zünkler: *Spracherkennung mit Hidden-Markov-Modellen unter Nutzung von unterscheidungsrelevanten Merkmalen*, Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, 1991

[Zwi82] E. Zwicker: *Psychoakustik*, Springer-Verlag, Berlin, 1982