

# **Sprachgesteuerte Signalwiedergabe**

Frank Kohler

Betreuer: Wolfgang Wokurek  
Institut für Maschinelle Sprachverarbeitung  
1997



## Zusammenfassung

In der vorliegenden Diplomarbeit wird eine Sprachgesteuerte Signalwiedergabe implementiert. Ziel ist es eine Steuerung eines virtuellen "Walkman" (CD-Player) durch Sprachkommandos zu entwickeln. Der Benutzer steuert durch die Aussprache von Kommandos in Form von Hauptsätzen die Wiedergabe von aufgenommenen Signalen, zum Beispiel: Spiele das dritte Lied, wiederhole den letzten Titel.

Als Methode werden stochastische Wissensbasen in einem HMM - basierten Spracherkenner integriert. Durch die stochastische Modellierung der semantischen Gliederung wird eine Robustheit bezüglich syntaktischen und semantischen Eingabefehlern erreicht.

## **Vorwort**

Die vorliegende Arbeit entstand während der Zeit meiner Diplomarbeit an dem Institut für maschinelle Sprachverarbeitung an der Universität Stuttgart.

An erster Stelle möchte ich mich bei meinem Betreuer Herrn Dr. Wolfgang Wokurek für die stete Betreuung dieser Arbeit und für sein offenes Ohr für auftauchende Probleme bedanken.

Bei Herrn Johannes Müller von der Technische Universität München bedanke ich mich für die Informationen, welche er mir über sein Projekt zur Verfügung stellen konnte.

Des weiteren bedanke ich mich bei Markus Fach, der mich bei der Spracherkennung mit HTK sehr hilfreich unterstützten konnte.

Herrenberg und Stuttgart im Oktober 1997

Frank Kohler

---

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b><u>EINLEITUNG</u></b>  | <b>5</b>  |
| 1.1      | SPRACHVERARBEITUNG ZUR MENSCH-MASCHINE-KOMMUNIKATION                    | 5         |
| 1.2      | REPRÄSENTATIONSEBENEN EINER SPRACHLICHEN ÄUßERUNG                       | 7         |
| 1.2.1    | “BOTTOM-UP”- VERSUS “TOP-DOWN” - ANSATZ                                 | 9         |
| 1.2.2    | REGELBASIERTER VERSUS STATISCHER ANSATZ                                 | 11        |
| 1.3      | EINSCHRÄNKUNG AUF BEGRENZTE DOMÄNE                                      | 15        |
| 1.4      | VORSTELLUNG DES PROJEKTES   | 16        |
| <b>2</b> | <b><u>SPRACHERKENNUNG</u></b>   | <b>18</b> |
| 2.1      | HIDDEN MARKOV MODELL  | 18        |
| 2.2      | DAS HIDDEN MARKOV MODELL TOOLKIT  | 22        |
| 2.2.1    | PRÄPARATION DER DATEN   | 22        |
| 2.2.2    | TRAINIEREN DER DATEN  | 24        |
| 2.3      | BEOBACHTUNGSFOLGE   | 27        |
| <b>3</b> | <b><u>ANSÄTZE ZUR SEMANTISCHEN DECODIERUNG GESPROCHENER SPRACHE</u></b> | <b>29</b> |
| 3.1      | ZWEI- ODER MEHRSTUFIGER ANSATZ  | 29        |
| 3.2      | EINSTUFIGER ANSATZ  | 30        |
| 3.3      | MEHRSTUFIGER VERSUS EINSTUFIGER ANSATZ                                  | 30        |
| <b>4</b> | <b><u>DER SEMANTISCHE DECODER</u></b>                                   | <b>32</b> |
| 4.1      | WAS IST SEMANTIK  | 32        |
| 4.2      | DIE SEMANTISCHE GLIEDERUNG  | 34        |
| 4.3      | AUSWAHL DER TYPEN UND WERTE   | 39        |
| 4.3.1    | PRAGMATISCHER GESICHTSPUNKT   | 39        |
| 4.3.2    | LINGUISTISCHER GESICHTSPUNKT  | 39        |
| 4.3.3    | EIN BEISPIEL EINER SEMANTISCHEN GLIEDERUNG                              | 40        |
| <b>5</b> | <b><u>CD-PLAYER</u></b>   | <b>41</b> |
| 5.1      | ANFORDERUNGEN AN DIE UMSETZUNG  | 41        |
| 5.2      | BEFEHLSGENERATOR  | 41        |
| 5.2.1    | LÖSUNGSANSATZ   | 42        |

---

|           |                                |           |
|-----------|--------------------------------|-----------|
| 5.2.2     | PRÄPROZESSOR                   | 42        |
| 5.2.3     | DER ÜBERSETZER                 | 43        |
| 5.2.4     | DIE ANSTEUERUNG DES CD-PLAYERS | 45        |
| <b>6</b>  | <b>ÜBERBLICK</b>               | <b>47</b> |
| <b>7</b>  | <b>DISKUSSION UND AUSBLICK</b> | <b>49</b> |
| <b>8</b>  | <b>PROBLEME</b>                | <b>51</b> |
| <b>9</b>  | <b>ANHANG</b>                  | <b>53</b> |
| 9.1       | ABBILDUNGSVERZEICHNIS          | 53        |
| 9.2       | TYPEN- UND WERTEINVENTAR       | 53        |
| 9.3       | GRAMMATIK DES HTK              | 54        |
| 9.4       | VORBELEGUNG DER TYPEN          | 55        |
| 9.5       | CD - BEFEHLE                   | 55        |
| <b>10</b> | <b>LITERATUR</b>               | <b>57</b> |

# 1 Einleitung

---

## 1.1 Sprachverarbeitung zur Mensch-Maschine-Kommunikation

Spracherkennung, Sprachverstehen und Sprachübersetzung sind drei Disziplinen der automatischen Spracherkennung, welche für eine benutzeradäquate Mensch-Maschine-Kommunikation sehr wünschenswert wären, jedoch technisch und technologisch derzeit noch nicht hinreichend beherrscht werden. Natürliche Sprache ist das meistverwendete Kommunikationsmittel zwischen den Menschen, wird in der Regel von allen beherrscht, funktioniert ohne den Einsatz von Händen oder Füßen und ist ohne visuelle Rückkopplung und damit auch im dunkeln möglich. 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 obigen Disziplinen ausgeführt werden können.

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

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

$$\text{Spracherkennung: Sprache} \rightarrow W \quad (1.1)$$

Hierbei ist der Bedeutungsinhalt des gesprochenen oder geschriebenen Textes völlig irrelevant für die Reaktion des Programm. Ein Einzelworterkenner kann nur für sich losgelöste Worte erkennen. Dies kann auch insofern erweitert sein, das beim Diktieren eines zusammenhängenden Textes kurze Pausen zwischen den einzelnen Worten gemacht werden müssen. Anspruchsvoller sind Systeme zur Erkennung fließender Sprache. Dabei sind zwischen den Worten keine Pausen notwendig, wobei jedoch das Problem der Koartikulation zu bewältigen ist. Sprecherunabhängige Systeme sind nur auf die Ausspracheeigenheiten eines bestimmten Sprechers angepaßt, während sprecherunabhängige Systeme gesprochene Eingaben von verschiedenen 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 angeordneten Datenbankzugriff ausführen:

$$\text{Sprachverstehen: Sprache} \rightarrow I \quad (1.2)$$

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, des weiteren 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  $W_{\text{ziel}}$ ) definiert:

$$\text{Sprachübersetzung: Sprache} \rightarrow W_{\text{ziel}} \quad (1.3)$$

Analog zum Sprachverstehen kann die Umwandlung 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.

## 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. 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:

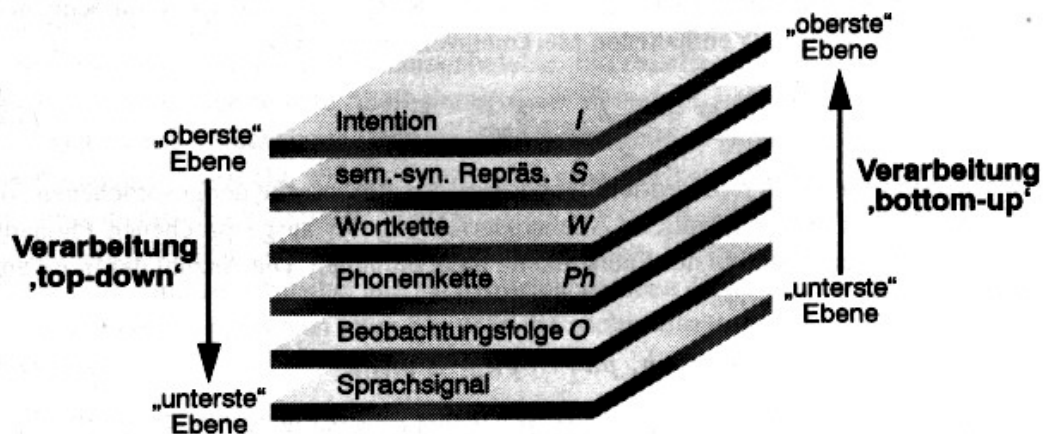


Abbildung 0.1: Repräsentationsebenen einer sprachlichen Äußerung

•

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.

- Die *Beobachtungsfolge* (oder Merkmalsvektorenfolge)  $O$  wird durch Vorverarbeitung des Sprachsignals gewonnen. Dazu werden sehr kurze, im vorliegenden Fall 25 ms dauernde Signalabschnitte 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.
- 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  $NP$ .
- 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  $NW$ .
- Als *semantisch-syntaktische Repräsentationsebene*  $S$  wird in dieser Arbeit die semantische Gliederung als formale Darstellung

eingeführt. Sie bildet bei der Verarbeitung der Wortkette zur Intention eine notwendige und zweckmäßige Zwischenebene. Je 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.

- Die *Intention*  $I$  gilt als “oberste” Ebene. Sie repräsentiert genau diejenigen applikationsspezifischen Kommandos, die am Rechner die vom Benutzer gewünschte Aktion auslösen oder die ihm gewünschte Information geben [Sta97].

### 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 “tieferere”:

- 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, da 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 gering wahrscheinliche Hypothesen werden verworfen und zu späteren Zeitpunkten nicht mehr betrachtet. 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 statischer 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 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.

$$P(\text{Ph}_i|O) = 1 \quad (1.4)$$

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.

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

$$I_E = \text{argmax } P(I|\text{Sprachsignal}) \quad (1.5)$$

Die Dimensionalität der Wahrscheinlichkeit  $P(I|\text{Sprachsignal})$  wäre viel zu hoch, um Gl. (1.5) direkt zu maximieren. So erscheint es sinnvoll, die vorher definierten Ebenen in obige Gleichung mit aufzunehmen.

$$P(I|\text{Sprachsignal}) = \underset{s \ o}{[P(I|S) * P(S|O) * P(O|\text{Sprachsignal})]} \quad (1.6)$$

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. (1.6) anstatt aller  $S$  und  $O$  nur das zum gesuchten  $I$  korrespondierende  $S$  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  $T_O$  des gegebenen Sprachsignals in eine

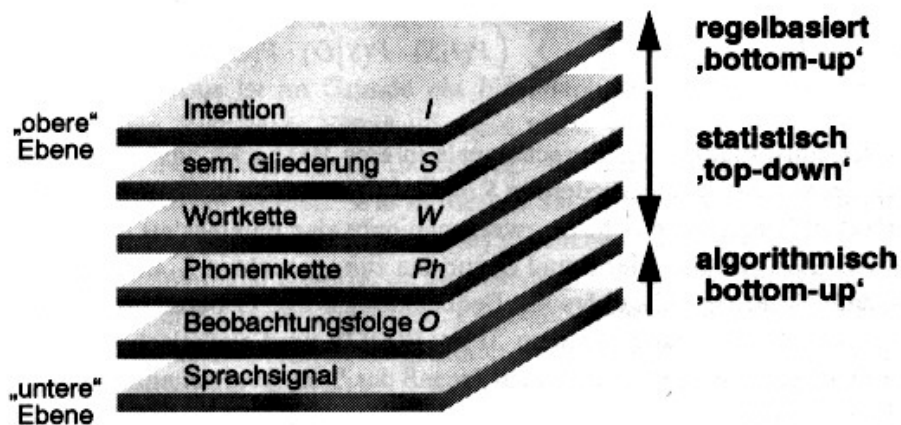
---

korrespondierende Beobachtungsfolge oder Merkmalsvektorenfolge  $O$ .

- Der semantische Decoder sucht mit stochastischen Methoden diejenige semantische Gliederung  $S$ , welche die bedingte Wahrscheinlichkeit für das Auftreten einer semantischen Gliederung gegeben eine Beobachtungsfolge  $O$  maximiert.
- Der Intentionsdecoder führt die regelbasierte Transformation  $T_I$  der semantischen Gliederung  $S$  in die für die Benutzerintention repräsentativen Maschinenbefehle aus.

Die Zusammenfassung der Module Vorverarbeitung, semantische Decodierung und Intentionsdecodierung wird im folgenden als Sprachverstehen bezeichnet.

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.



**Abbildung 0.2:** Mögliche Aufteilung der Verarbeitungsschritte

Das in obiger Abbildung 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 dargestellt abläuft.

**Abbildung 0.3:** Umwandlung von natürlicher Sprache in Computersprache (nach Pieraccini)

Ein Semantik Übersetzer (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 semantisch Decoder kann für eine N-L-nahe formale semantische Sprache S-L relativ einfach gehalten werden. 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 [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 “Walkman” entworfen und vollständig implementiert. Mit dieser Applikation ist es möglich, mit

gesprochenen Anweisungen einen CD-Player zu bedienen, d.h. einzelne Befehle auszuführen.

#### **1.4 *Vorstellung des Projektes***

Das Projekt Natürlichsprachliche Mensch - CD - Schnittstelle wurde in mehreren Teilschritten bearbeitet.

Als erstes wird aus der natürlichsprachlichen Eingabe mittels Hidden Markov Modell Toolkit aus trainierten Daten eine Beobachtungsfolge  $O$  extrahiert, um aus dieser Beobachtungsfolge mit einem semantischen Decoder die semantische Gliederung  $S$  zu erhalten. Aus der semantischen Gliederung  $S$  werden über den Befehlsgenerator die CD - Befehle erstellt. Diese CD - Befehle steuern dann den virtuellen Walkman oder CD-Player. Der theoretische Ablauf ist in der folgenden Grafik dargestellt.

**Abbildung 0.4:** Natürlichsprachliche Mensch - CD - Schnittstelle

---

## 2 Spracherkennung

---

Die Spracherkennung erfolgte mittels des Hidden Markov Model Toolkit welches zur Sprachaufnahme und Trainings der Daten benutzt wurde.

Es wurde die Version HTK 2.0 benutzt.

In diesem Kapitel soll zunächst einmal das Hidden Markov Modell vorgestellt, und außerdem der Ablauf und die Funktionsweise des Hidden Markov Model Toolkit beschrieben werden.

### 2.1 *Hidden Markov Modell*

Was sind Hidden Markov Modelle und was ist das Hidden Markov Model Toolkit?

In diesem Abschnitt wird beschrieben, was Hidden Markov Modelle sind und wie man sie zur Erkennung einsetzen kann.

#### **Statistische Modellierung und die Regel von Bayes**

Spracherkennung durch HMMs basiert auf der Idee der statistischen Modellierung. Äußerlich betrachtet, soll ausgehend von dem Schallereignis die wahrscheinlichste Wortfolge berechnet werden. Gegeben ist eine Äußerung  $O$ , ist es wahrscheinlicher, dass sie das Wort  $A$  oder das Wort  $B$  enthält? Diese Frage ist so direkt nicht einfach zu beantworten. Bei der Statistischen Modellierung wendet man deshalb einen Trick an, indem man das Problem gewissermaßen umkehrt. Das Problem würde sich dann in zwei Fragen verwandeln: Wenn angenommen wird es handelt sich um das Wort  $A$ , wie wahrscheinlich ist dann das konkret gemessene Schallereignis  $O$ ? Wie wahrscheinlich ist  $O$ , wenn angenommen wird, es handelt sich um das Wort  $B$ ? Die beiden Arten, das Erkennungsproblem zu formulieren, stehen in einem mathematischen Zusammenhang, den die Bayessche Regel beschreibt:

$$P(A_i|B) = P(A_i)P(B|A_i) / P(B) \quad (2.1)$$

Voraussetzung für die Regel von Bayes ist  $\sum_i P(A_i) = 1$  und  $\forall_{i \neq j} A_i \cap A_j = \emptyset$ . Durch Anwenden der Bayesschen Regel kann man bedingte Wahrscheinlichkeiten “umdrehen”, also  $P(A_i|B)$  durch  $P(B|A_i)$  ausdrücken.

Der Nenner der Formel,  $P(B)$ , gibt an, wie wahrscheinlich die Beobachtung überhaupt ist. Diese Größe ist nur von der Messung und nicht davon abhängig, ob Wort A oder Wort B geäußert wurde. Da sie für alle  $A_i$  gleich ist, und es nur interessiert, welches  $P(A_i|B)$  am größten ist, wird der Nenner bei Erkennen oft gar nicht berücksichtigt.

### **Was sind Hidden Markov Modelle?**

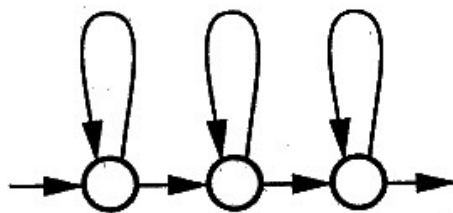
HMMs sind endliche Automaten, denen zwei stochastische Prozesse unterlegt sind. Wie im vorherigen Abschnitt bereits angesprochen, sollen die HMMs modellieren, wie das Sprachsignal für einen bestimmten Laut, ein bestimmtes Wort oder Satz aussieht. Man kann sich also vorstellen, das HMM erzeuge dieses Sprachsignal (genauer: daraus berechnete Merkmalsvektoren). Außerdem soll mit ihnen die Wahrscheinlichkeit für eine bestimmte Folge von konkret observierten Daten angegeben werden können, also die Wahrscheinlichkeit, dass ein bestimmtes HMM eine bestimmte Folge von Daten erzeugt hat.

### **HMMs für die Spracherkennung**

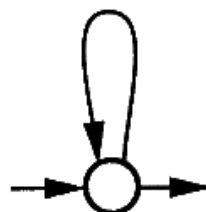
In der Spracherkennung wird meist nicht direkt das Sprachsignal (Zeitsignal) als Ausgabe der HMMs aufgefaßt, sondern daraus berechnete Merkmalsvektoren. Dazu wird für kleine Ausschnitte aus dem Zeitsignal (typisch 25,6 ms) eine Frequenzanalyse durchgeführt. Diese Analyse wird mit festem Abstand (typisch 10 ms) wiederholt über das ganze Sprachsignal immer wieder durchgeführt. Statt der eindimensionalen Wertefolge des Zeitsignals erhält man dadurch eine Folge von hochdimensionalen Vektoren (typische Dimension 10-40). Die Idee dabei ist, dass bei den Merkmalsvektoren die Unterschiede zwischen den zu unterscheidenden Klassen erhöht werden, aber die

Unterschiede bei Mustern eine Klasse gegenüber der Betrachtung des Zeitsignals verringert werden. Gegenüber dem oben gezeigten HMM ändert sich nicht viel: Es werden eben statt eines einzigen Zufallswertes gleich mehrere Zufallswerte (Zufallsvektor) ausgegeben.

In der Spracherkennung werden meist links-rechts-Modelle verwendet, das heißt, die Modelle können nur ganz kleine Zyklen enthalten, bei denen das Modell im selben Zustand bleibt. Der Zustand kann nie zu einem Zustand wechseln, in dem der Automat bereits zu einem früheren Zeitpunkt war. Man kann diese Modelle somit so zeichnen, dass alle Pfeile von links nach rechts gehen. Bei Modellen für Phoneme werden häufig 3 Zustände verwandt. Der erste soll den vom vorausgehenden Phonem beeinflussten Abschnitt, der zweite den stabilen Abschnitt in der Mitte, der dritte Zustand den vom folgenden Phonem beeinflussten Abschnitt beschreiben (s. Abb. 2.1). Sprechpausen werden häufig mit nur einem Zustand modelliert (s. Abb. 2.2). Wenn man HMMs für größere Einheiten, etwa Wörter, verwenden möchte, ist es sinnvoll, die Anzahl der Zustände zu erhöhen.



**Abbildung 0.1:** links-rechts-Modell



**Abbildung 0.2:** Sprechpause



---

## **Zusammenfassung: Allgemeiner Aufbau von HMM-basierten Spracherkennungssystemen**

HMM-Spracherkennungssysteme haben meist folgenden Aufbau: Die ankommenden Schallwellen werden an der Membran eines Mikrophons in elektrische Schwingungen umgewandelt und anschließend digitalisiert oder liegen bereits in digitalisierter Form vor. Aus dem digitalisierten Sprachsignal, dem sog. Zeitsignal, werden Merkmalsvektoren berechnet. Für verschiedene mögliche Wörter oder Wortfolgen wird die Wahrscheinlichkeit dafür errechnet, dass die vorliegenden Merkmalsvektoren von den Modellen der vom Erkennen in Betracht gezogenen Wörter erzeugt wurden. Unter Anwendung der Bayesschen Regel kann man errechnen, welches Modell eines Wortes oder einer Wortfolge am wahrscheinlichsten der Folge von Merkmalsvektoren entspricht. Die Möglichkeit mit der höchsten Wahrscheinlichkeit wird als erkanntes Wort bzw. erkannte Wortfolge ausgegeben [Lab97].

### **2.2 *Das Hidden Markov Modell Toolkit***

Das Hidden Markov Modell Toolkit besteht aus verschiedenen Teilschritten, welche nacheinander abgearbeitet werden, diese werden in den folgenden Kapiteln beschrieben, mit Schwerpunkten auf HInit, HRest und HVite.

#### **2.2.1 Präparation der Daten**

Als ersten Schritt in einem Erkennen Entwicklungsprojekt ist die Vorbereitung der Daten. Denn diese Sprachdaten werden zum Trainieren und Testen benötigt.

##### **Die Grammatik**

Zuerst wurde die Grammatik für die möglichen Sätze entwickelt, bei der Grammatik sollte es nicht darauf ankommen, alle möglichen Fehler zu erkennen, sondern möglichst viele Sätze zuzulassen, um alle möglichen Kombinationen von Eingaben zu ermöglichen. Einige Beispiele sollen jetzt angeführt werden:

spiele / stop  
springe zum Titel Nummer fünf  
spiele das Lied Nummer eins, lauter  
wiederhole das erste Lied, leise  
werde etwas lauter  
spiele von 2 Minuten 30 Sekunden bis 5 Minuten 10 Sekunden  
wiederhole die letzten zehn Sekunden

Es soll mit dieser Grammatik möglich sein einfache Befehle wie spiele, stop oder wiederhole einzugeben, aber auch erweiterte Kombinationen mit Angabe des Titels oder auch der Angabe von Start- und Endezeiten in welchen gespielt werden soll.

Ein weiteres Kriterium ist die Lautstärke, es soll über die Eingabe auch möglich sein die Lautstärke zu steuern, d.h. lauter oder leiser zu werden, oder mit einer bestimmten Lautstärke zu spielen.

Außerdem soll es möglich sein, dass irgendwelche Floskeln wie "bitte", "etwas" an beliebiger Stelle eingegeben werden können, ohne die Funktion bei der Ausführung zu beeinflussen

Ein Problem, dass bei dieser Art von Grammatik entsteht ist folgendes, es ist durch die freie Eingabe sehr wohl möglich Grammatikalisch falsche oder auch semantisch total unsinnige Sätze einzugeben, da aber davon ausgegangen werden kann, dass der Anwender nicht versucht das System mit "idiotischen" Eingaben zu überlisten scheint es unnötig die Grammatik so aufzublähen, dass sie alle möglichen Eingaben erkennt und nur die Grammatikalisch richtigen zulässt.

Die hierfür entwickelte Grammatik ist im Anhang beigefügt.

### **Das Lexikon**

Bei der Erstellung des Lexikons wurden fünf Klassen von Wörtern unterschieden, um eine geeignete Auswahl der benötigten Worte treffen zu können.

Als erstes die Klasse *Befehle* welche die benötigten Befehle oder Kommandos für die Steuerung des CD-Players beinhalten, wie zum Beispiel "spiele", "springe", "wiederhole" usw..

Als nächstes die Klasse *Objekte*, welche in zwei Unterklassen eingeteilt wird, nämlich die *Titelobjekte* die für die Steuerung notwendigen Titel enthält wie “Lied”, “Song”, “Titel”, und die *Zeitobjekte* welche die Objekte der Zeitangaben wie “Sekunden”, “Minuten” oder “Stunden” enthält.

Die Klasse der *Positionen* enthält wiederum zwei Unterklassen nämlich die Klasse der *Positionord*, welche die Ordinalzahlen beinhaltet und die Klasse *Positionzahl*, welche die Zahlen beinhaltet. Die Ordnungszahlen sind von eins bis zehn in allen möglichen Kombinationen, “ein”, “eine”, “einer”, “eines”, “erste”, “erster”, “erstes” vorhanden, und die Cardinalzahlen sind ebenfalls von eins bis zehn und danach in Zehnerschritten bis einhundert verfügbar.

Die nächste Klasse ist die Klasse für die *Lautstärke* in welcher die Lautstärken “leise”, “mittel” und “laut” mit ihren Derivationen enthalten sind.

Abschließend gibt es noch die Klasse *Ergänzungsworte* in welcher die Klasse Artikel und Sonstige mit den sogenannte Floskeln und Höflichkeitsworten, wie “bitte”, “danke” usw., enthalten sind.

Im folgenden Schritt muß jetzt ein Lexikon erstellt werden, welches nicht nur die Wörter, sondern auch die zugehörigen Phoneme enthält. Diese wurden aus dem bekannten Lexikon CELEX herausgefiltert. Dies sollte eigentlich mit HDMAN geschehen, war aber wegen der Größe von CELEX leider nicht möglich, so dass es mit einem Zusatzprogramm erstellt werden mußte, dieses Lexikon enthält nun alle eingetragenen Wörter mit den zugehörigen Phonemen.

## 2.2.2 Trainieren der Daten

Für die Durchführung des Trainings ist die Erstellung der Prototypen für die Sprachlaute und damit für die Modellierung der initialen Hidden Markov Modelle unbedingt erforderlich. Der Prototyp eines jeden Sprachlautes spezifiziert das Hidden Markov Modell für den jeweiligen Sprachlaut. Aus diesem Grund ist hierauf ein besonderes Augenmerk zu legen.

---

Im folgenden wird in kurzer Form der Ablauf des Trainings und des Testens unter Zuhilfenahme des Hidden Markov Modell Toolkit beschrieben.

Das Training gliedert sich in drei Schritte, die durch die Werkzeuge HInit, HRest, HERest des Hidden Markov Modell Toolkit unterstützt werden.

**HInit:**

Dieses Hidden Markov Modell Tool wird zur Initialisierung der einzelnen Hidden Markov Modelle verwendet. Dabei wird eine Menge von Beobachtungssequenzen herangezogen um initiale Abschätzungen für die Übergangswahrscheinlichkeiten und die Standardabweichungen zu errechnen. HInit kann für die initialen Schätzungen der phonembasierten Spracherkennung der grundlegenden Hidden Markov Modelle eingesetzt werden. Die Beobachtungssequenzen bestehen dabei aus Sequenzen des gesprochenen Trainingsmaterials. HInit schneidet sich die Labels und damit Variationen des jeweiligen Hidden Markov Modells aus den Trainingsdaten aufgrund ihrer Vorgaben heraus.

Eingabe für HInit sind die Prototypendefinitionen der Hidden Markov Modelle der Sprachlaute, die die erforderliche Topologie der Hidden Markov Modelle definiert, sowie die Label- und Datafiles des Trainingsmaterials.

**HRest:**

HRest führt eine grundlegende Baum-Welch Rückwärtsabschätzung der Parameter der einzelnen Hidden Markov Modelle ebenfalls unter Zuhilfenahme einer Menge von Beobachtungssequenzen durch. HRest arbeitet dabei identisch zu HInit in bezug auf die Auswahl der Segmente aus den Trainingsdaten. Als Eingabe erhält HRest die bereits initialisierten Modelle von HInit und die jeweiligen Label- und Datefiles.

**HERest:**

HERest wird verwendet, um ein einzelnes Rückabschätzen der Parameter aus HRest mit Hilfe des Embedded-Training von Baum-

---

Welch durchzuführen. Dabei kann die Zahl der Wiederholungen dieser Trainingsphase vom Benutzer vorgegeben werden.

**HVite:**

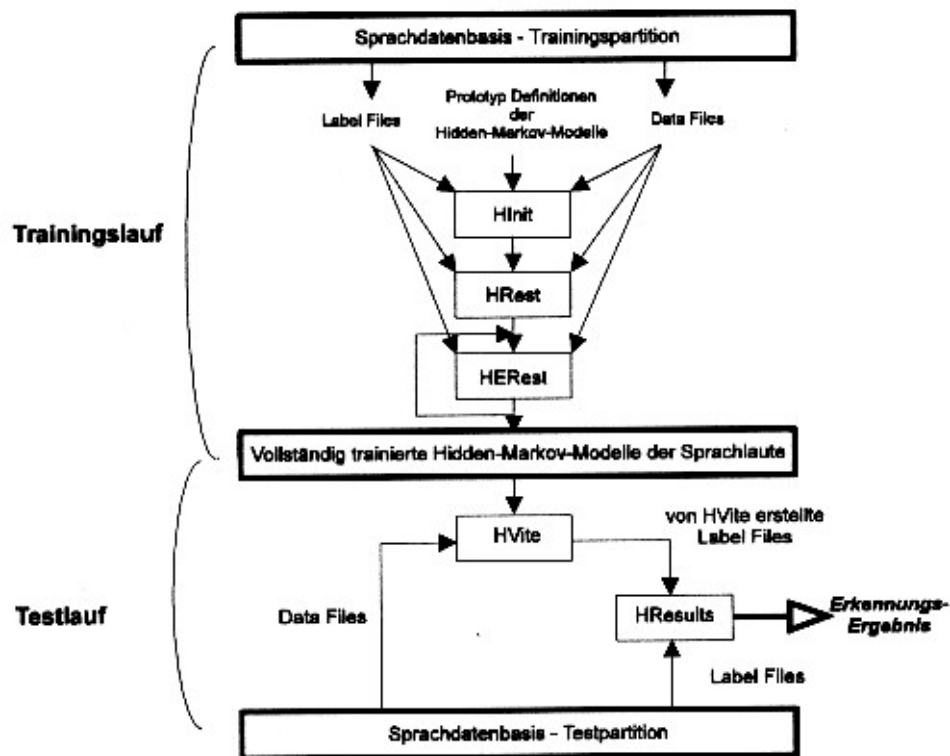
Diese Tool ist ein Viterbi-Erkennen. Er vergleicht ein ihm vorgegebenes Netzwerk von Hidden Markov Modellen mit einem oder mehreren Testfiles und gibt eine Übersetzung in Hidden Markov Modell Toolkit Format für jedes gelabelte Testfile aus. Als Eingabe benötigt HVite das Netzwerk sowie die Testfiles und die trainierten Hidden Markov Modelle der Sprachlaute.

**HResults:**

diese Werkzeug ist zuständig für die Auswertung der erkannten Modelle. Es liest eine Menge von Hidden Markov Modell Toolkit Labelfiles und vergleicht diese mit den zugehörigen von HVite übersetzten Dateien (Rec-Files). HResults unterteilt dabei die Auswertung in eine Satzweise Analyse und eine phonemweise Auswertung.

Um einen tieferen Einblick in die Arbeitsweise des Hidden Markov Modell Toolkit zu gewinnen verweise ich auf das Hidden Markov Modell Toolkit Reference-Manual.

Die folgende Grafik stellt nochmals den Zusammenhang der Werkzeuge des Hidden Markov Modell Toolkit dar:



**Abbildung 0.3:** Zusammenhang der Hidden Markov Modell Toolkit Tools für Training und Test

Bei HVite besteht die Möglichkeit die Übergangswahrscheinlichkeiten der Hidden Markov Modelle durch einen Parameter “p” (fixed transition log probability) so zu verändern, dass ein Einfluß auf die Erkennung der Modelle ausgeübt werden kann [HTK].

### 2.3 Beobachtungsfolge

Die Beobachtungsfolge ist ein wichtiger Teil zwischen der Ausgabe des Hidden Markov Modell Toolkit und der Eingabe in den semantischen Decoder, sie ist praktisch das Brückenstück zwischen diesen beiden Teilen.

Die Beobachtungsfolge besteht aus einer Kette von Worten, welche später von dem semantischen Decoder ausgewertet werden. Diese Wortkette wird unsortiert und nicht bearbeitet von dem Hidden Markov Modell Toolkit weitergegeben. Die Daten des Hidden

Markov Modell Toolkit werden in einer Datei gespeichert, welche so an den semantischen Decoder weitergegeben werden.

### 3 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 diese eine jedermann vertraute Repräsentationsebene von natürlicher Sprache darstellt. Dieses Vorgehen hat einen zweistufigen Ansatz, bestehend aus Spracherkennung und semantischer Textanalyse, zur Folge.

**Abbildung 0.1:** Zwei- bzw. einstufiger Ansatz zur semantischen Decodierung von Sprache

#### 3.1 *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, das ist ein Netzwerk mit dazwischenliegenden Übergängen. Die nachfolgende Textanalyse analysiert dies und generiert hieraus 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 [Mue97].

### **3.2 *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 Wissensbasen, 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 auf ein nachfolgendes davon losgelöstes Modul auswirken.

### **3.3 *Mehrstufiger versus einstufiger Ansatz***

Der meist verbreitete Ansatz zur semantischen Decodierung einer Beobachtungsfolge ist ein zweistufiger Ansatz mit der Wortebene als Schnittstelle, zwischen den hierbei eingesetzten Modulen Wortketten-Decodierung und semantische Analyse wird grundsätzlich Bottom-Up vorgegangen. Wird jedoch bei der Wortketten-Decodierung ein Fehler begangen, kann dieser von der nachfolgenden semantischen Textanalyse nur in wenigen Fällen korrigiert werden. Außerdem steht bei der Wortketten-Decodierung 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 Wortketten-Decoders 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 Phonem-Decoder, Phonem-Graphem Umsetzer 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.

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

Aus diesen Gründen liegt ein einstufiger Ansatz auf der Hand, bei dem alle zur Verfügung stehenden Wissensbasen mit semantischem, syntaktischem, morphologischem, phonetischem und akustischem Wissen gleichzeitig und konsistent zusammenwirken können. Somit kann eine aus den multiplizierten Wahrscheinlichkeiten aller Wissensbasen bestehende Verbundwahrscheinlichkeit über alle Repräsentationsebenen gebildet werden, die gemeinsam, Top-Down und innerhalb eines einzigen Rechenschrittes maximiert werden kann. 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 Rechnerkapazität 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 ist ausführlich in [STA96] beschrieben.

---

## 4 Der semantische Decoder

---

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 oder akustische Information 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.

### 4.1 *Was ist Semantik*

Unter dem Begriff Semantik versteht man im allgemeinen die Bedeutung von Zeichen oder Zeichenfolgen. Im folgenden seien damit ausschließlich sprachliche Zeichen 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.

Die akustische Vorstellung ist keine laut ausgesprochenen 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. Selbst wenn die vom Gehör aufgenommenen 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 Disambiguierung 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 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 bisher 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 kann 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 CD-Player oder einer Fahrplaninformation eben nur Äußerungen innerhalb der jeweiligen Domäne zu erwarten sind.

## 4.2 ***Die semantische Gliederung***

Zum Einsatz innerhalb einer sprachverstehenden Applikation wird eine formale Repräsentation des Bedeutungsinhaltes einer natürlichsprachlichen gesprochenen oder geschriebenen Äußerung gesucht, die

1. wortnah genug ist, um einen direkten oder probabilistischen Bezug zur jeweiligen Wortkette zu ermöglichen
2. hierarchisch aufgebaut ist, um eine Struktur nach festen (auch rekursiven) Regeln zu ermöglichen und um bestehende Abhängigkeit probabilistisch zu erfassen
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
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 begrenzten Domäne stammt, und keinen Nebensatz aufweist, eingeführt [Mül94]. Sie kann aufgefaßt werden als eine baumartige und damit hierarchische Struktur, die sich aus  $N$  kleineren bedeutungstragenden Einheiten zusammensetzt, welche im folgenden semantische Untereinheiten (kurz Semune)  $s_n$  genannt werden:

$$S = \{s_1, s_2, \dots, s_n, \dots, s_N\} \quad (4.1)$$

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

$$q_1[s_n], \dots, q_x[s_n], \dots, q_X[s_n] \in \{s_{n+1}, \dots, s_N, \text{leer}\} \quad (4.2)$$

beschrieben werden:

$$s_n = (t[s_n], v[s_n], q_1[s_n], \dots, q_X[s_n]) \\ \text{mit } X \geq 1 \quad (4.3)$$

- Der Typ  $t[s_n]$  gibt die Anzahl  $X$  der Nachfolger fest vor und schränkt die Menge möglicher Typen  $t[q_1[s_n]], \dots, t[q_X[s_n]]$  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

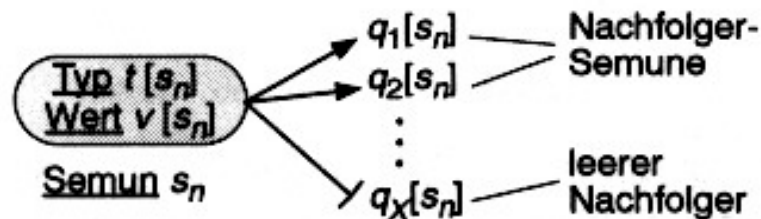
$$q_x[s_n] \in \{s_{n+1}, \dots, s_N\} \quad (4.4)$$

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

$$q_x[s_n] = \text{leer} \quad (4.5)$$

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

Ein Semun  $s_n$  mit den Verweisen auf  $X$  Nachfolger 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 Nachfolge Semun wird durch die Kante " $\rightarrow$ " markiert. Im Gegensatz kennzeichnet die Kante " $-|$ " den Verweis auf einen leeren Nachfolger.



**Abbildung 0.1:** Darstellung eines Semuns  $s_n$  mit  $X$  Nachfolgern

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

Ein Semun  $s_n \in S$  mit rekursiv betrachtet allen seinen Nachfolgern bildet einen Ast  $S(s_n)$ . Die Semune innerhalb eines Astes sind fortlaufend von  $n$  beginnend indiziert.

$$S(s_n) = \{s_n, s_{n+1}, s_{n+2}, \dots\} \quad (4.6)$$

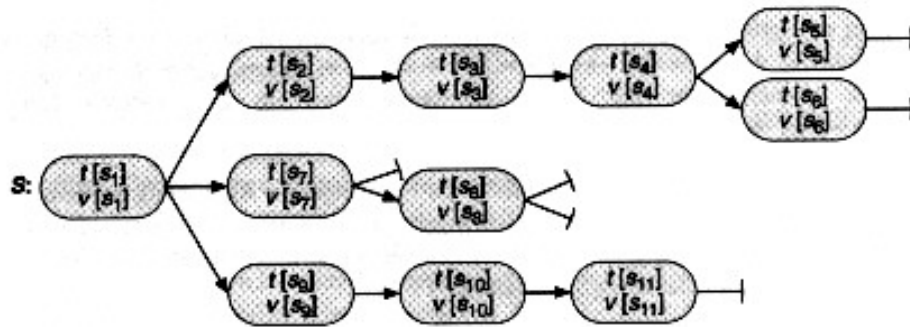
Jeder Ast bildet somit eine Teilmenge der semantischen Gliederung

$$S(s_n) \subset S \text{ für } n \neq 1 \quad (4.7)$$

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

$$S(s_1) = S \quad (4.8)$$

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



**Abbildung 0.2:** Verknüpfung mehrerer Sememe 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 Sememe (Verweise auf leere Nachfolger zählen ebenfalls). Der Ast  $S(s_2) = \{s_2, s_3, s_4, s_5, s_6\}$  würde aus fünf Sememen, 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 Sememen bestehen. Jedes Semem repräsentiert einen Teil der Bedeutung der gesamten Äußerung. Die Bedeutung eines Semems 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.

Eine Linie ist eine direkte Abfolge von  $m+1$  Sememen  $s_n, \dots, s_{n+m}$  innerhalb eines Astes mit jeweils genau  $X=1$  Nachfolgern. In der obigen Abbildung existieren demnach zwei solcher Linien, nämlich  $s_2, s_3$  und  $s_9, s_{10}, s_{11}$ .

Des weiteren soll festgestellt werden, dass die Anordnung der Sememe innerhalb einer Linie auf den von der semantischen Gliederung ausgewiesenen Bedeutungsinhaltes keinen Einfluß hat - im Klartext:

Die Anordnung der Sememe innerhalb einer Linie ist wahlfrei!

In obiger semantischer Gliederung S wäre zum Beispiel die Anordnung der Sememe  $s_2$  und  $s_3$  beliebig, das heißt entweder  $\rightarrow s_2 \rightarrow$

$s_3 \rightarrow$  oder  $\rightarrow s_3 \rightarrow s_2 \rightarrow$ . Entsprechendes gilt ebenfalls für die Semune-Anordnung von  $\rightarrow s_9 \rightarrow s_{10} \rightarrow s_{11} \text{---}$ .

Zwei semantische Gliederungen gelten als äquivalent, wenn sie die gleiche Information beinhalten. Eine äquivalente semantische Gliederung entsteht, durch Permutationen einzelner Semune innerhalb einer Linie.

Die semantische Gliederung bietet den Vorteil, sie ist eine wortnahe Darstellung des Bedeutungsinhaltes und ermöglicht eine unmittelbare Beziehung zu einer Wortkette  $W$ . Sie berücksichtigt lediglich den semantischen Inhalt der Äußerung ohne Berücksichtigung der pragmatischen Konstellation.

### 4.3 *Auswahl der Typen und Werte*

Ausschließlich bei der Festlegung der Typen und Werte sowie deren Beziehungen zur Wortkette steckt im beschriebenen Ansatz zur Semanticdecodierung pragmatisches (domänenspezifisches) und linguistisches (sprachspezifisches) Wissen.

#### 4.3.1 **Pragmatischer Gesichtspunkt**

Die sinnvolle Auswahl der Typen und Werte geschieht zunächst heuristisch nach pragmatischen Gesichtspunkten. Die als semantische Gliederung darzustellenden Äußerungen sollen ausschließlich aus einer umgrenzten Domäne stammen. Es muß daher ein geeignetes Typen- und Wertinventar gebildet werden, welches nur den Bedeutungsinhalt von zu erwartenden Äußerungen abdeckt. Für Äußerungen außerhalb der betrachtenden Domäne sind keine Typen und Werte zu definieren (Es wird von einem sprachverstehenden CD-Player nicht verlangt werden etwas anderes als CD's abzuspielen).

#### 4.3.2 **Linguistischer Gesichtspunkt**

Um den Suchraum bei der semantischen Decodierung klein zu halten und um mögliche Fehlinterpretationen zu vermeiden, sollte das semantische Modell möglichst wenig Hypothesen bilden. Das kann

dadurch eingeschränkt werden, indem die Vielfalt möglicher Nachfolger eines bestimmten Typs gering gehalten wird.

### 4.3.3 Ein Beispiel einer semantischen Gliederung

Hier wird das Beispiel “Spiele das erste Lied, laut” in der semantischen Gliederung dargestellt.

**Abbildung 0.3:** Eine semantische Gliederung “Spiele das erste Wort, laut”

Zum Beschreiben des Sprachmaterials werden vier Typen mit ca. 100 Worten eingesetzt, im Anhang wird das benutzte Typen- und Wertinventar aufgelistet.

---

## 5 CD-Player

---

Nach der Dekodierung der semantischen Gliederung  $S$  generiert der Befehlsgenerator im nächsten Schritt den korrespondierenden CD-Befehl. Dieser verständliche CD-Befehl, der sich an die definierte Befehlsstruktur hält, wird schließlich dem CD-Player übermittelt.

### 5.1 *Anforderungen an die Umsetzung*

Die Umsetzung der semantischen Gliederung in einen CD-Befehl ist nicht immer ohne weiteres möglich. Es lassen sich verschiedene vom Nominalablauf abweichende Situationen erkennen:

1. Die von Decoder generierte semantische Gliederung ist nicht sinnvoll
2. Es fehlen für den Befehl wichtige Daten, d.h. die semantische Gliederung ist zwar korrekt aber unvollständig
3. Die in der semantischen Gliederung spezifizierten Daten, wie zum Beispiel Lieder sind dem Gesamtsystem nicht bekannt

Faßt man die verschiedenen Situationen zusammen, so lassen sich die folgenden Anforderungen an den Befehlsgenerator ableiten:

- Übersetzung der semantischen Gliederung in CD-Befehle
- Validierung der Befehle auf Plausibilität und Vollständigkeit
- Einbindung aktueller Umgebungsinformation und des CD-Playerzustandes
- Dialoggeführtes Einbeziehen des Bedieners zur Vervollständigung der Befehle unter Ausnutzung der zur Verfügung stehenden Ressourcen

### 5.2 *Befehlsgenerator*

Da die semantische Gliederung als wortnahe Darstellung einer Äußerung stark kontextbehaftet ist und die Umgebungskonstellation sowie vorherige Äußerungen nicht einbezieht, 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.

Als triviales Beispiel sei die Wortkette “spiele lauter” betrachtet. Das Wissen, mit welcher Lautstärke vorher gespielt wurde ist der semantischen Gliederung nicht bekannt. Weiterhin kann es bei der Äußerung “spiele das 15. Lied” zu Problemen kommen, falls ein 15. Lied gar nicht vorhanden ist. Der Befehlsgenerator verbindet nun die semantische Gliederung mit der aktuellen Umgebungskonstellation und erzeugt daraus eine lineare Abfolge von interpretierbaren Befehlen.

### 5.2.1 **Lösungsansatz**

Für die Umsetzung einer semantischen Gliederung in eine den Wünschen des Benutzers entsprechende Ausgabe wird eine Kombination aus einem Präprozessor und einem Compiler vorgeschlagen. Ein ähnlicher Aufbau hatte sich bereits innerhalb eines Systems zum Verstehen natürlicher Sprache bewährt [Hab80].

**Abbildung 0.1:** Blockdiagramm des Befehlsgenerators

### 5.2.2 **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].

### **Abbildung 0.2:** Blockdiagramm eines relationalen Präprozessors

Durch den Präprozessor werden folgende Funktionen durchgeführt:

#### **Zusammenführung:**

Die semantische Gliederung beinhaltet oftmals verschiedene Semune die Effektiv den gleichen Wert  $t[s_n]$  (die gleiche Bedeutung) haben, wie zum Beispiel "spiele" oder "play". Diese Semune werden in der Zusammenführung zusammengefaßt damit die Anzahl der verschiedenen Werte  $t[s_n]$  auf ein Minimum reduziert werden kann.

#### **Optimieren:**

Zur Einsparung von Rechenleistung, werden solche Semune eliminiert, die innerhalb der semantischen Gliederung keine Information beisteuern.

#### **Vorbelegung:**

Wenn Typen  $t[s_n]$  keinen Wert  $v[s_n]$  beinhalten werden hierfür vorbestimmte Werte den Typen zugeordnet. Diese vorbestimmten Werte können im Anhang nachgelesen werden.

## **5.2.3 Der Übersetzer**

Vorher wurde immer von einem Compiler gesprochen, es gibt aber zwei Möglichkeiten dies zu realisieren, erstens über den Compiler und zweitens über einen Interpreter. Also der Übersetzer kann entweder ein Compiler oder aber ein Interpreter sein, im

nachfolgenden sollen die Unterschiede und Vor- und Nachteile etwas erläutert werden.

### **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 hat den Vorteil, dass seine Ausführung direkt auf Eingaben des Benutzers reagieren kann.

**Der Compiler:**

Eine oder mehrere vorverarbeitete semantische Gliederungen bilden das Programm in der Quellsprache, für den Compiler. Diese Eingabe kann nicht direkt umgesetzt werden, da der Rechner präzise und eindeutige Anweisungen benötigt.

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.*

#### 5.2.4 Die Ansteuerung des CD-Players

Wie oben schon angeführt wurde für die Ansteuerung des CD-Players der Compiler gewählt, da bei der Applikation nur mit wenigen Rückfragen gerechnet werden muß, und da ein äquivalentes Ausführungsprogramm den Anforderungen entspricht.

Für den CD-Player erweisen sich die folgenden vier Zustände als notwendig und zweckmäßig:

- Der Zustand "Befehl" ist der Startzustand der Gliederung
- Der Zustand "Objekt" beinhaltet die verschiedenen Objekte
- Der Zustand "Position" bestimmt die aktuelle Position
- Der Zustand "Laut" ist für die Veränderung der aktuellen Lautstärke verantwortlich

Die Ansteuerung des CD-Players setzt die von der Vorverarbeitung aktualisierten Daten voraus. Im Rahmen der vorliegenden Arbeit wurde die von C++ bereitgestellte Bibliothek cdaudio.h benutzt. Jedes einzelne Semun der Daten wird eingelesen und die nötigen Befehle zur Ansteuerung des CD-Players nacheinander ausgeführt.

**CD-Befehle:**

Für die Ansteuerung des CD-Players gibt es verschiedene Befehle die dieser ausführen kann, diese Befehle sind im Anhang vorgestellt und erläutert.

Es gibt darüber hinaus auch noch einige weitere Befehle, welche allerdings zur Ansteuerung eines CD-Players nicht notwendig oder aber auch nicht zu benutzen sind, deshalb sollen in diesem Rahmen nur die benötigten oder benutzbaren Befehle angesprochen werden, sollten weitere Befehle notwendig sein muß auf die Literatur von C++ verwiesen werden, wo diese Befehle angeführt und auch beschrieben sind.

## 6 Überblick

Abschließend soll das ganze System noch einmal als Einheit vorgestellt und die Zusammenhänge der einzelnen Teile kurz erklärt werden.

### **Abbildung 0.1:** Ein Gesamtüberblick über das System

Der Ablauf des Programmes geschieht in mehreren Schritten hintereinander, auf diese Schritte wird im Nachfolgenden näher eingegangen.

#### **1.Schritt:**

Als erstes erfolgt die natürlichsprachliche Eingabe der Worte von einem Menschen, diese Worte werden über ein Mikrofon eingegeben und werden als Sprachsignal an das Hidden Markov Modell Toolkit weitergegeben.

#### **2.Schritt:**

Hier werden die Worte von dem Spracherkenner verarbeitet der sie mit den trainierten Daten der Hidden Markov Modelle vergleicht.

**3.Schritt:**

Um dann in diesem Schritt die Beobachtungsfolge zu erstellen, die das Verbindungsstück zwischen dem Hidden Markov Modell Toolkit und dem semantischen Decoder darstellt.

**4.Schritt:**

Dieser Teil stellt den semantischen Decoder dar. Der semantische Decoder bearbeitet die Beobachtungsfolge indem er die erkannten Worte auswertet. Die Auswertung erfolgt durch die Erstellung der semantischen Gliederung, welche die vier Typen mit den dafür verantwortlichen Werten beinhaltet. Die Werte werden den bestimmten Typen zugeordnet, um für die nachfolgenden Schritte eine möglichst einfache Weiterverarbeitung zu ermöglichen. Die vier Typen der semantische Gliederung sind zuerst die Kommandos, dann die Objekte, die Position und die Lautstärke.

**5.Schritt:**

Der 5. Schritt ist die semantische Gliederung, welche die Übergabe des semantischen Decoders an den Befehlsgenerator bildet.

**6.Schritt:**

Der Befehlsgenerator ist in zwei Teile unterteilt, den Präprozessor und den Übersetzer, der Präprozessor arbeitet die Daten für den Übersetzer auf, damit dieser eine möglichst einfache Arbeitsweise hat. In diesem Schritt werden auch die Typen die keine Werte aus der Eingabe erhalten vorbelegt. Um dann im nächsten Schritt den Übersetzer auszuführen.

**7.Schritt:**

Der letzte Arbeitsgang wird vom Übersetzer (Compiler) geleistet, er liefert die Ausgabe an den CD-Player, hier wird der CD-Player mit den für ihn typischen Befehlen angesteuert, um dann die eingelegte CD abzuspielen oder an eine bestimmte Stelle der CD zu springen und diesen Titel anzuspielden.

## 7 Diskussion und Ausblick

Im Rahmen dieser Arbeit konnte ein sprachverstehendes System entworfen werden und für die Domäne “virtueller Walkman” erfolgreich implementiert werden. Die Demonstrationsapplikation zeigt den Einsatz von natürlicher gesprochener Sprache zur Steuerung eines laufenden Programms.

Ein Kernpunkt der vorliegenden Arbeit war der Entwurf der semantischen Gliederung als Repräsentation des Bedeutungsinhaltes einer sprachlichen Äußerung. Ihr Aufbau kombiniert semantisch-syntaktische Konzepte miteinander und erlaubt die Darstellung vieler Semantikstrukture in einer begrenzten Domäne.

Ein wichtiges Kriterium in Richtung uneingeschränkter Spracheingabe ist die Verringerung der “Out of Vocabulary” - Rate durch eine Vergrößerung des dem System zur Verfügung stehenden Vokabulars. Eine übliche Methode ist dabei das Sammeln von weiterem Trainingsmaterial, was jedoch einen zeitintensiven und un kreativen Prozeß darstellt.

Eine ausschließlich sprachliche Eingabe ist für viele Anwendungsgebiete keine vollkommen zufriedenstellende Lösung. Für die Anwendung bei einem CD-Player reicht die Spracheingabe jedoch vollständig aus, allerdings kann bei anderen Anwendungen wie einem Grafikeditor wie NASGRA [Sta97] diese Eingabe nicht ausreichen. Bei diesen Anwendungen wäre eine andere Art der Eingabe wie eine direkt Koordinateneingabe sinnvoll, es wäre wünschenswert diese Eingabe direkt über Berührungen (Touchscreen) oder indirekt über mit der Maus ausführen zu können. Auch bei der Korrektur falscher Eingaben ist es sinnvoll mit bereitgestellten Eingabemodi die Erkennungsrate und die Benutzerakzeptanz zu erhöhen. Eine Erweiterung eines informationsverarbeitenden Systems auf multimodale Eingabe hätte nicht nur 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 Benutzer bereits in den Entwicklungsprozeß eingebunden wird. Die Qualität eines technischen Systems sollte im Zuge diese Vorgehens nicht nur nach einer möglichst großen Funktionalität gemessen werden, sondern es sollten dazu auch subjektive, psychologische und ergonomische Werte Berücksichtigung finden.

---

## 8 Probleme

---

Bei der Entwicklung des sprachgesteuerten CD-Players traten einige Probleme auf, auf die im folgenden eingegangen werden soll, um mögliche Probleme zu erklären und eventuell für zukünftige Anwendungen Anregungen zu geben, dass diese Fehler nicht noch einmal gemacht werden oder diese Probleme frühzeitig erkannt werden können.

### **Spracherkennung:**

Die Spracherkennung hat einen gravierenden Nachteil, sollte jemand versuchen irgendwelche nicht vorhandenen Kommandos oder auch Kommandos die nicht sinnvoll sind einzugeben, so wird der Präprozessor diese ohne Fehlermeldung abarbeiten, er legt dann allerdings vorbestimmte Werte fest, so dass dann immer ab dem ersten Lied, mit einer bestimmten Lautstärke, gespielt wird.

### **Semantische Decoder:**

Bei der semantischen Decodierung trat folgendes Problem auf: da die Zahlen zwei verschiedene Arten von Bedeutungen haben konnten, hätte, bei der Eingabe von folgenden Sätzen: "Spiele das dritte Lied" und "Spiele noch drei Lieder" zu Verwechslungen führen können.

Aus diesem Grund konnte die zweite Möglichkeit nicht betrachtet werden, da hier die Doppelbedeutung der Zahlen nicht unterschieden werden konnte.

### **CD-Player:**

Die Ansteuerung des CD-Players erwies sich als sehr kompliziert, da die vorhandene Library nicht korrekt eingebunden werden konnte, bei dieser Library mußten Veränderungen vorgenommen werden, welche sich auf mögliche Erweiterungen negativ auswirken können. Es wurde allerdings nicht getestet ob und wie sich die Veränderungen bei einer eventuellen Änderung auswirken.

Außerdem gab es noch folgendes Problem bei der Ansteuerung des CD-Players, es war sehr schwierig, überhaupt Literatur über eine mögliche Software mäßige Ansteuerung des CD-Players zu

beschaffen. Es würde nur in der Library von C++ Information über diese Ansteuerung gefunden, welche allerdings auch nicht ausreichend genug ist, um darauf aufbauen zu können.

## 9 Anhang

---

### 9.1 *Abbildungsverzeichnis*

|   |    |
|---|----|
| Abbildung 1.1: Repräsentationsebenen einer sprachlichen Äußerung                                  | 8  |
| Abbildung 1.2: Mögliche Aufteilung der Verarbeitungsschritte .....                                | 17 |
| Abbildung 1.3: Umwandlung von natürlicher Sprache in<br>Computersprache (nach Pieraccini) .....   | 18 |
| Abbildung 1.4: Natürlichsprachliche Mensch - CD - Schnittstelle....                               | 20 |
| Abbildung 2.1: links-rechts-Modell.....   | 24 |
| Abbildung 2.2: Sprechpause .....  | 24 |
| Abbildung 2.3: Zusammenhang der Hidden Markov Modell Toolkit<br>Tools für Training und Test ..... | 32 |
| Abbildung 3.1: Zwei- bzw. einstufiger Ansatz zur semantischen<br>Decodierung von Sprache .....    | 34 |
| Abbildung 4.1: Darstellung eines Semuns $s_n$ mit X Nachfolgern .....                             | 44 |
| Abbildung 4.2: Verknüpfung mehrerer Semune zum Baum einer<br>semantischen Gliederung S .....      | 45 |
| Abbildung 4.3: Eine semantische Gliederung "Spiele das erste Wort,<br>laut" .....                 | 48 |
| Abbildung 5.1: Blockdiagramm des Befehlsgenerators .....  | 51 |
| Abbildung 5.2: Blockdiagramm eines relationalen Präprozessors.....                                | 52 |
| Abbildung 6.1: Ein Gesamtüberblick über das System.....   | 57 |

### 9.2 *Typen- und Wertinventar*

Die vier benutzten Typen mit ihren zugehörigen Werten, in Klammern die derzeitige Anzahl

#### **Kommando (6):**

spiele, springe, wiederhole, werde, play, repeat

#### **Objekt (6):**

Lied, Song, Titel, Stück, Sekunden, Minuten

**Position (62):**

eins, ein, eine, eines, einer, erste, erstes, ersten,  
zwei, zweite, zweites, zweiten, zweiter,  
drei, dritte, drittes, dritten, dritter,

....

zehn, zehnte, zehntes, zehnten, zehnter,  
zwanzig, dreißig, vierzig, fünfzig, sechzig, siebzig, achtzig, neunzig,  
hundert

**Laut (5):**

leise, mittel, laut, leiser ,lauter

**9.3 Grammatik des HTK**

\$kommando = spiele | springe | werde | wiederhole | play | stop;

\$titelobjekt = lied | song | titel | stueck;

\$zeitobjekt = sekunden | minuten;

\$positionord = erste | zweite | dritte | vierte | fuenfte |  
sechste | siebte | achte | neunte | zehnte | ersten | zweiten | dritten |  
vierten | fuenften | sechsten | siebten | achten | neunten | zehnten |  
erstes | zweites | drittes | viertes | fuenftes | sechstes | siebtes | achttes |  
neuntes | zehntes | erster | zweiter | dritter | vierter | fuenfter | sechster |  
siebter | achter | neunter | zehnter | letzte | naechste | aktuelle;

\$positionzahl = eins | zwei | drei | vier | fuenf | sechs | sieben |  
acht | neun | zehn | zwanzig | dreissig | vierzig | fuenfzig | sechzig |  
siebzig | achtzig | neunzig | hundert;

\$lautstaerke = leise | leiser | mittel | laut | lauter;

\$artikel = der | die | das | zum | zur | dem | den | bis | von;

\$sonstiges = schnell | etwas | ganz;

```
(start
([bitte] $kommando [bitte]
  (
    ($artikel $titelobjekt [nummer] $positionzahl) \
    # spiele das Lied Nummer eins
  | ($artikel $positionord $titelobjekt) \
    # spiele das erste Lied
  | ([ $sonstiges] [ $lautstaerke]) \
    # spiele | stop
  | ($artikel $positionzahl $zeitobjekt [ $positionzahl] \
    [ $artikel $positionzahl $zeitobjekt [ $positionzahl]])
    # spiele von ... bis
  )
[bitte])
ende)
```

## 9.4 *Vorbelegung der Typen*

Für die vier Typen wurden folgende Vorbelegungen definiert:

- Befehl: Spiele
- Objekt: Lied
- Position: eins
- Lautstärke: mittel

## 9.5 *CD - Befehle*

- CDopen: öffnet einen bestimmten CD-Player
- CDplay : spielt die CD, beginnt am Anfang und spielt die ganze CD
- CDplaytrack: spielt einen track und endet wenn der track vorüber ist
- CDplaytrackabs: spielt einen track, startet bei einer Zeiteingabe und endet am Ende der tracks

- CDplayabs: startet bei einer Zeitangabe und spielt die ganze CD
- CDreadda: liest digitale audio Daten
- CDbestreadsize: gibt Daten des CD-Players zurück
- CDseek: fährt an eine bestimmte Stelle um digital audio Daten zu lesen
- CDseekblock: fährt zu einer bestimmten Blocknummer
- CDseektrack: fährt an einen bestimmten track
- CDstop: hält den CD-Player an
- CDeject: wirft die CD aus
- CDallowremoval: Schält den Eject-Button ein
- CDpreventremoval: Schält den Eject-Button aus
- CDclose: schließt einen bestimmten CD-Player
- CDgetstatus: liefert aktuelle Informationen über den Status zurück
- CDtogglepause: ändert den aktuellen Zustand der Pause, wenn Pause gedrückt ist dann beginne zu spielen und andersherum
- CDgettrackinfo: liefert die Information über den aktuellen track Status
- CDsetvolume: ändert die aktuellen Lautstärke
- CDgetvolume: liefert Information über die aktuelle Lautstärke

---

## 10 Literatur

---

- [Aho88] A.V. Aho, R. Sethi, J.D. Ullmann: Compilerbau - Teil 1, Addison-Wesley, Bonn, 1988
- [Bro95] M. Brockhaus, A. Ertl: Übersetzerbau, Vorlesungsskript, Institut für Computersprachen, Technische Universität Wien 1995
- [Fis97] C. Fischer, P. Havel, G. Schmidt: Kommandierung eines Serviceroboters mit natürlicher, gesprochener Sprache,
- [Hab80] C.U. Habel, C.R. Rollinger, A. Schmidt, H.J. Schneider: A Logic-Oriented Approach to Automatic Text Understanding, Natural Based Computer System, Carl Hanser, München, 1980
- [HTK] S. Young, J. Jansen, J. Odell, D. Ollason, P. Woodland: The HTK BOOK, Reference Manual zu HTK 2.0
- [Lab97] W. Wokurek, S. Rapp: Laborübung Spracherkennung
- [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
- [Mül95] J. Müller, H. Stahl: Stochastic Modelling of Syntax and Semantics, Tagungsband KI 1995 Activities: Workshops, Posters, Demos (Bielefeld, Deutschland), S. 229-230
- [Mue97] J. Müller: Die semantische Gliederung zur Repräsentation des Bedeutungsinhaltes innerhalb sprachverstehender Systeme, Herbert Utz, München, 1997

- [Pie93] R. Pieraccini, E. Levin: Learning How to Understand Language, Tagungsband Eurospeech 1993 (Berlin, Deutschland), S.1407-1412
- [Sta96] H. Stahl, J. Müller, M. Lang: An efficient Top-Down Parsing Algorithmus for Understanding Speech by Using Stochastic Syntactic and Semantic Models, Tagungsband ICASSP 1996, (Atlanta, USA), S.397-400

## **Erklärung**

Ich, Frank Kohler, versichere, dass diese Arbeit von mir selbständig erstellt und dafür nur die angegebenen Hilfsmittel verwendet wurden.

Die schriftliche Ausarbeitung erfolgte mit der Textverarbeitung Word für Windows Version 7.0.

Unterschrift