Dialektsynthese

Erweiterungen an einem Sprachsynthesesystem

Diplomarbeit
bei Prof. Dr.-Ing. Hansjörg Mixdorff

vorgelegt von Markus Binsteiner
am Fachbereich Informatik der
Technischen Fachhochschule Berlin

Berlin, 30. Juni 2004

Inhaltsverzeichnis

1. Einleitung **1**

I. Theoretischer Teil **2**

2. Fachliches Umfeld **3**
2.1. Phonetische Grundlagen 3
2.1.1. Das *SAMPA*-Alphabet 3
2.1.2. Die menschliche Lautproduktion 4
2.2. Text-To-Speech System 6
2.2.1. Vorverarbeitung 7
2.2.2. Graphem-Phonem-Umsetzung 8
2.2.3. Prosodiegenerierung 10
2.2.4. Akustische Synthese 12
2.3. Das Synthesesystem *MBROLA* 15
2.3.1. *PSOLA* 16

3. Der bairische Dialekt **18**
3.1. Die Laute 19
3.1.1. Vokale 19
3.1.2. Diphthonge 23
3.1.3. Konsonanten 25
3.2. Bairische Eigenarten 27
3.2.1. Entrundung 27
3.2.2. Umlaute 28
3.2.3. Konsonantenschwächung 28
3.3. Bairische Grammatik 29
3.3.1. Das Substantiv 29

3.3.2. Die Artikel 29
3.3.3. Das Wortgeschlecht 30
3.3.4. Singular und Plural 30
3.3.5. Weitere Unterschiede 31

II. Praktischer Teil 32

4. Anbindung von *MBROLA* an *TFHTTS* 33
4.1. Die *MBROLA*-Bibliothek „`mbrplay.dll`“ 33
4.2. Die Anpassung von *TFHTTS* 34
4.2.1. Die Oberfläche `eval_tts.exe` 34
4.2.2. Die Bibliothek `tfhtts.dll` 37

5. Erstellen des *MBROLA*-Inventars 39
5.1. Vorbereitung der Phoneme und Diphone 39
5.2. Aufnahme 42
5.2.1. Ort 42
5.2.2. Material 42
5.2.3. Testlauf 43
5.2.4. Durchführung 44
5.3. Segmentieren 45

6. Erstellung der Regeln für die Umsetzung 49
6.1. Der Regelsatz 49
6.1.1. Wörter 49
6.1.2. Zahlen 52
6.2. Das Wörterbuch 52

7. Probleme / Einschränkungen 54
7.1. Grammatik / Wortschatz 54
7.2. Prosodie 55

8. Schlusswort 56

A. SAMPA-Liste der deutschen Laute 57

B. Die Klasse `MbrolaTranslation` 58

C. Das Perl-Skript `translate.pl` 62

D. Praat-Skript als Segmentierhilfe 68

1. Einleitung

Das Thema dieser Diplomarbeit ist die Erweiterung des bestehenden Sprachsynthesesystems *TFHTTS*, welches an der TFH Berlin von Herrn Prof. Dr.-Ing. Mixdorff betreut wird. Dieses Sprachsynthesesystem ermöglicht es, in den Sprachen Englisch, Französisch, Italienisch und Deutsch, per Tastatur eingegebenen Text in eine akustische Sprachausgabe umzusetzen. Die Erweiterung des Systems gliedert sich in zwei Teilaufgaben:

- Zum Einen ist es nötig, *TFHTTS* durch eine programmiertechnische Implementierung an *MBROLA*, einen frei erhältlicher Sprachsynthesizer, anzubinden, und dadurch die ursprüngliche Syntheseeinheit in *TFHTTS* zu ersetzen.
- Zum Anderen soll im Anschluss *TFHTTS* dahingehend erweitert werden, dass es möglich ist, eine dialektal gefärbte, bairische Stimme als Sprachausgabe zu erhalten.

Letzteres lässt sich entweder durch Ersetzen bzw. Nachbilden der bairischen Laute mit schon Vorhandenen, Hochdeutschen realisieren. Oder aber man erstellt ein neues Inventar für *MBROLA*, welches dann die Eigen- und Besonderheiten der dialektalen Aussprache genau abbilden kann. Für diese Möglichkeit habe ich mich im vorliegenden Fall entschieden, weil einige Laute und Lautkombinationen des Bairischen nur sehr unzureichend nachgebildet werden können. Dazu jedoch später mehr. Abschliessend muss ein Regelsatz erstellt werden, der angibt, wie welche Wortteile von geschriebener hochdeutscher Sprache in gesprochenen bairischen Dialekt umgewandelt werden sollen.

Ein Wort noch zur Schreibweise, bzw. dem Unterschied der Schreibweise „bairisch“ – „Bayern“. Bayern wurde bis zum Anfang des 19. Jahrhunderts tatsächlich „Baiern“ geschrieben. Doch dann entdeckte König Ludwig I. seine Begeisterung für die antike griechische Kultur und ließ als Reminiszenz das „i“ durch den griechischen Buchstaben „y“ im Namen des von ihm regierten Landes ersetzen. Da sich die Sprachwissenschaft jedoch mit der Sprache des Volksstammes der Baiern befasst, ist es auch heute noch korrekt diese „Bairisch“ zu nennen. Deswegen werde ich im Folgenden alle Bezüge, die auf die Sprache hinweisen mit „i“ schreiben, und solche, die das Landesgebiet meinen, mit „y“.

Teil I.

Theoretischer Teil

2. Fachliches Umfeld

2.1. Phonetische Grundlagen

Die Phonetik ist die Lehre der von Menschen hervorgebrachten Laute und sie beschäftigt sich mit den beobachtbaren Eigenschaften derselben (vgl. [Gib98]). Diese Eigenschaften können mithilfe der modernen Computertechnologie sehr gut beschrieben und visualisiert werden. Untersucht wird in der Phonetik, welche Laute in verschiedenen Sprachen existieren, wie diese Laute erzeugt werden, wie sie übertragen werden und auch, wie sie vom menschlichen Ohr wieder empfangen und verarbeitet werden. Für diese Arbeit ist im Wesentlichen interessant, wie Laute erzeugt (und miteinander kombiniert) werden.

2.1.1. Das *SAMPA*-Alphabet

Es existiert eine international genormte Lautschrift, IPA (Internationales Phonetisches Alphabet), welche versucht, für alle auf der Welt verwendeten Laute eine symbolische Repräsentation bereitzustellen. Das Alphabet wurde zwar auch in den *Unicode*-Standard aufgenommen, hat sich im IT-Bereich allerdings nicht durchgesetzt. Hier wird meist mit einer auf 7-Bit-ASCII-Code basierten Darstellung von Lauten mit dem Namen *SAMPA* (Speech Assessment Methods Phonetic Alphabet) oder der Erweiterung X-SAMPA gearbeitet[1]. In *SAMPA* wird z. B. die lautliche Repräsentation des als „*Ä*“ bekannten Buchstaben[2] mit dem Symbol „`E`“ realisiert.

Ein besonderer Laut ist die Stille, welche mit dem Zeichen „`_`“ beschrieben wird. Sie kennzeichnet Sprechpausen und muss, z. B. bei der Produktion eines Diphoninventars, als Laut angesehen werden. Eine Auflistung der gängigen deutschen Laute mit ihren *SAMPA*-Repräsentationen findet sich im Anhang A auf Seite 57.

[1] nähere Informationen zu *SAMPA* unter `http://www.phon.ucl.ac.uk/home/sampa/home.htm`
[2] in H̲ä̲me, K̲ä̲se, der gleiche Laut taucht aber z. B. auch in He̲rr oder Me̲sser auf

Auch das für diese Arbeit erzeugte *MBROLA*-Inventar benützt eine am *SAMPA*-Alphabet angelehnte Notation.

2.1.2. Die menschliche Lautproduktion

Die Erzeugung von Lauten, die von Anderen verstanden werden sollen, beginnt in der Lunge. Hier wird durch das Zwerchfell ein Luftstrom erzeugt, welcher unterhalb der Glottis[3] einen Luftdruck aufbaut. Durch diesen Luftdruck im Zusammenspiel mit sich stetig öffnenden und schließenden Stimmlippen wird in der Folge ein schwingender Luftstrom aufgebaut, der vom Sprecher gesteuert werden kann. Dieser Luftstrom ist gewissermaßen die Basis der späteren Sprache, er enthält ein ganzes Frequenzspektrum. Durch die Reso-

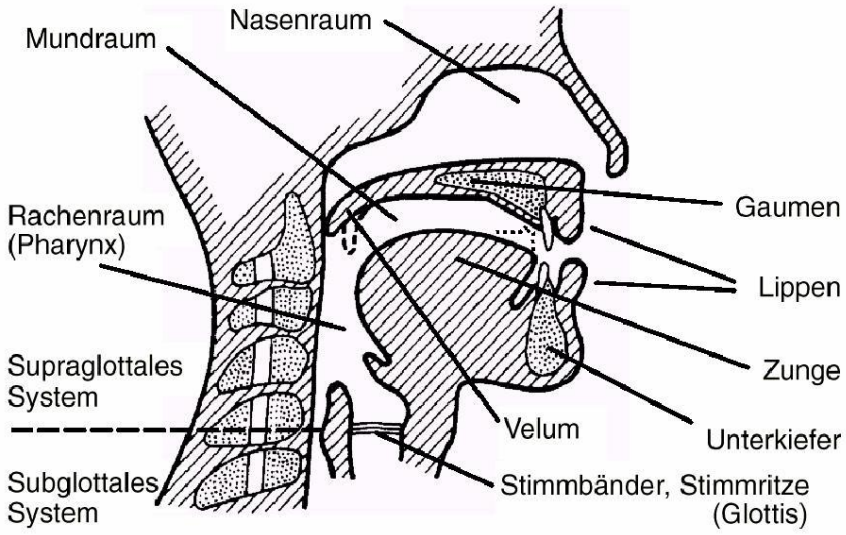


Abbildung 2.1.: Stimmbänder und Vokaltrakt

nanzräume des Vokaltraktes werden anschließend bestimmte Frequenzen verstärkt und andere gedämpft. Hier beginnt das Formen des später erzeugten Klangbildes. Der veränderte Luftstrom wird abschließend durch den Vorgang, den man als „Artikulation“ kennt, unter anderem durch die Zunge, die Lippen, den Gaumen, die Zähne usw. gehemmt und somit modelliert, was zur Entstehung von Vokalen und Konsonanten führt.

[3] der stimmbildende Teil des Kehlkopfes

Vokale

Kann der Luftstrom weitgehend ungehemmt den Körper verlassen, entstehen Vokale. Der Klang eines Vokals wird größtenteils durch seine Formanten bestimmt.

Formanten kennzeichnen Frequenzbereiche bei der Aussprache eines vokalischen Lautes, welche durch Resonanzen verstärkt werden. Diese Resonanzen entstehen in den oben bereits erwähnten Resonanzräumen im Vokaltrakt. Im *Spektrogramm* des erzeugten Klanges werden die Formanten als dunkele Bereiche des Frequenzspektrums dargestellt, da diese Frequenzen eine höhere Energiedichte aufweisen. Abbildung 2.2 zeigt die ersten vier Formanten des bairischen Lautes `i:` (die Formanten werden zur Verdeutlichung zusätzlich noch durch eine weiße Linie gekennzeichnet).

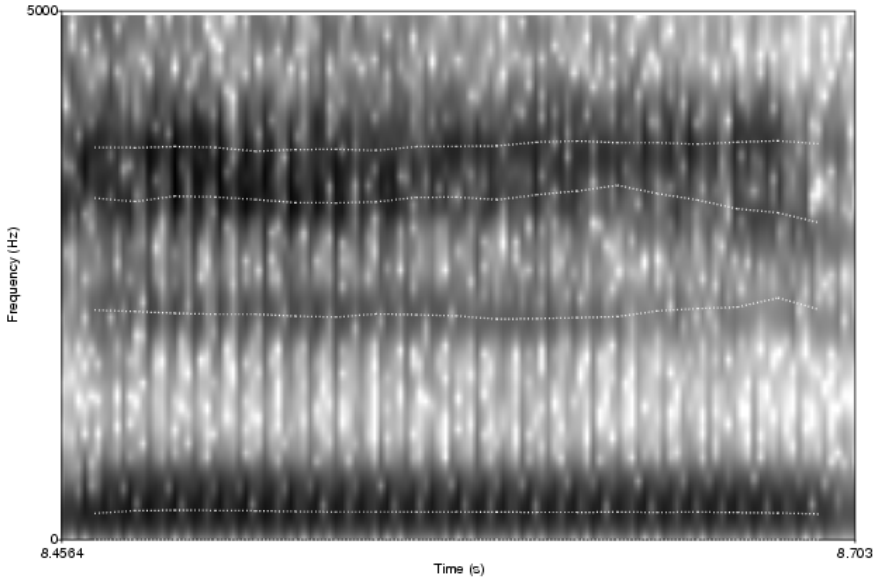


Abbildung 2.2.: Spektrogramm des Phonems `i:`

Jeder Vokal hat, bedingt durch die Veränderung des Vokaltraktes beim Sprechen, seine eigenen spezifischen Formantpositionen, abhängig bloß von kleinen Schwankungen, die von Sprecher zu Sprecher variieren. Ansonsten wäre ein Verstehen von Sprache nicht möglich. Die ersten beiden Formanten repräsentieren die Art des Lautes und werden benutzt, um vokalische Laute voneinander zu unterscheiden. Sie können durch eine Veränderung der Zungenstellung, in geringerem Maße auch durch die Rundung der Lippen und durch Nasalierung, bewusst vom Sprecher erzeugt werden. Anders als die höheren Formanten,

welche die Unverwechselbarkeit der Stimme eines Sprechers ausmachen und nicht von ihm gesteuert werden können, da sie von den jeweiligen körperlichen Eigenheiten des Sprechers abhängen.

Konsonanten

Konsonanten weisen im Gegensatz zu Vokalen aperiodische Schwingungen auf, welche durch Verengungen des Stimmtraktes entstehen. Die Luft kann dadurch nicht, wie oben beschrieben, ungehindert nach draussen. Es entstehen hörbare Turbulenzen. Konsonanten werden deshalb eher als Geräusch denn als Klang wahrgenommen. Man unterteilt Konsonanten im Allgemeinen anhand ihrer Stimmhaftigkeit, ihres Artikulationsortes und ihrer Artikulationsart.

Die Stimmhaftigkeit von Konsonanten teilt diese in Stimmhafte (b, d, g, l, m, n, N, R, v, z, Z) und Stimmlose (C, f, k, p, s, S, t, x). Ob ein Laut stimmhaft ist, hängt davon ab, ob sich die Glottis beim Sprechen periodisch öffnet und schließt.

Die Aufteilung nach Artikulationsort geschieht in die (hier relevanten) Kategorien *labial* (b, f, m, p, v), *alveolar* (d, j, l, n, s, t, z), *postveolar* (S), *palatal* (C, j), *velar* (g, N, R[4], x) und *glottal* (h).

Unter Artikulationsarten versteht man die Kategorien *Plosive* (b, d, g, k, p, t), *Frikative* (C, f, h, r, s, z, S, v, x, Z) und *Sonoranten* (l, j, m, n, N). Außerdem existiert noch die Gruppe der Affrikaten (pf, ts, tS).

2.2. Text-To-Speech System

TFHTTS[5], die Software, die für diese Arbeit erweitert wird, ist ein so genanntes Text-To-Speech (TTS) System. Unter Text-To-Speech System versteht man eine Applikation, welche geschriebenen Text in gesprochene Sprache wandeln kann. Sie besteht im Allgemeinen aus zwei Einheiten: Eine, welche den Text analysiert und in eine *symbolisch linguistische Repräsentation* wandelt, und eine Zweite, welche diese Repräsentation zu akustischen Signalen verarbeitet, die akustische Synthese. TFHTTS verfügt über eine Möglichkeit, Protokolle

[4] eigentlich *uvular*, gemeint ist das hochdeutsche „r“, nicht das bairische, welches viel weiter vorne gebildet wird

[5] eine Abbildung des Frontends `eval_tts` findet sich auf Seite 35

nach den einzelnen Verarbeitungsschritten auszugeben. Diese lassen sich sehr gut benutzen, um die Vorgänge in einem *TTS*-System zu veranschaulichen, was ich im Folgenden kurz versuchen will. Als Beispiel verwende ich die beiden Sätze *„Deutschland ist im Jahr 1974 Fußballweltmeister geworden. Wie alt warst du zu der Zeit?“*.

2.2.1. Vorverarbeitung

Die Analyse muss vorbereitend den Text in zusammengehörige Einheiten wie Sätze, Teilsätze und Wörter aufteilen und entsprechende Markierungen setzen. Außerdem wird hier die Interpretation von *Homographen*, Abkürzungen und Zahlen erledigt. Je nach Kontext[6] muss z. B. 1945 entweder als Eins-Neun-Vier-Fünf oder Neunzehnhundertvierundfünfzig gelesen werden. Unser Beispielsatz sieht nach der Vorverarbeitung folgendermaßen aus:

```
{Utterance:begin}{ClauseType:final}
#deutschland
#ist
#im
#{WClass:Nom}jahr
#{Zahl:year}1900
#{Zahl:integer}74
##{WClass:Nom}fußballweltmeister
#geworden
#*.{Sentence:end}
{Utterance:end}#@@
{Utterance:begin}{ClauseType:quest}
#wie
#alt
#warst
#du
#zu
#der
#{WClass:Nom}zeit
#*?{Sentence:end}
{Utterance:end}#@@
```

[6] etwa Telefonnummer oder Jahreszahl

Wie man sehen kann, wird erkannt, dass es sich um zwei getrennte Äußerungen handelt, wobei die Zweite anhand des Fragezeichens richtig als Frage gedeutet wird. Außerdem wird durch das Schlüsselwort *„Jahr“* die nachfolgende Zahl als Jahreszahl interpretiert und sofort in die zwei zu sprechenden Teile *Neunzehnhundert* sowie *Vierundsiebzig* aufgeteilt.

2.2.2. Graphem-Phonem-Umsetzung

Anschließend durchläuft das Ergebnis die sog. Graphem-Phonem-Umsetzung. Es wird entschieden, in welche *Phoneme* die *Grapheme* umgesetzt werden. Hierfür existieren zwei verschiedene Ansätze.

Es gibt die wörterbuchbasierte Umsetzung, welche Einträge eines tabellenartigen Wörterbuchs, das zu jedem Wort die lautliche Entsprechung bereithält, nach dem umzusetzenden Wort absucht und es dann mit dem Wörterbucheintrag ersetzt.

Und es gibt den regelbasierten Ansatz. Dieser versucht anhand von Regeln, Wortteile oder Buchstaben in die resultierenden Phoneme umzuwandeln (siehe dazu auch Kap. 6.1, Seite 49).

Welchen Ansatz man wählt, hängt stark von der zu synthetisierenden Sprache ab. Im Spanischen, wo die Aussprache anhand von Regeln sehr gut vorausgesagt werden kann, würde man wohl eher eine regelbasierte Umsetzung wählen, was im Englischen wiederum nur sehr bedingt gelingen kann, weil dort die Aussprache unter anderem sehr vom Kontext des Wortes abhängig ist[7]. In der Praxis wird aber fast immer eine Kombination beider Varianten gewählt. Wörter, die man nicht im Wörterbuch findet, werden anhand von Regeln umgesetzt. Das reduziert einerseits die Mühe bei der Wörterbucherstellung und den Speicherbedarf des Wörterbuchs, da nicht alle Wörter einer Sprache aufgenommen zu werden brauchen, andererseits deckt man doch den gesamten Wortschatz ab.

TFHTTS verfolgt ebenfalls eine kombinierte Strategie zur Bestimmung der richtigen Laute. Als Erstes wird in einem, ca. 5100 Wörter umfassenden, kleinen Lexikon gesucht. Dieses Lexikon deckt die am häufigsten verwendeten deutschen Wörter ab, enthält neben der lautlichen Repräsentation auch noch die Wortart des Begriffs und lässt sich sehr schnell durchsuchen. Findet sich das gesuchte Wort dort nicht, wird die Suche im großen Wörterbuch fortgesetzt. Das besteht aus ca. 300.000 Einträgen und ist als indizierte Datenbank angelegt, um die Suche zu beschleunigen. Für die vorliegende Arbeit wird das große Wör-

[7] z. B. die Betonung im Satz „To insert an insert.“

terbuch jedoch nicht verwendet, weil es zu viel Zeit beansprucht hätte, allen 300.000 sich darin befindlichen Wörtern eine bairische Aussprache zuzuordnen.

Schlägt auch diese Abfrage fehl, dann wird das Wort anhand der erstellten Regeln übersetzt. Das beinhaltet auch eine Generierung der Betonungsmarken.

Das Programm *TFHTTS* gibt uns nach der Wörterbuchsuche folgende Ausgabe:

```
Wort im kleinen Lexikon gefunden: deutschland
Wort im kleinen Lexikon gefunden: ist
Wort im kleinen Lexikon gefunden: im
Wort im kleinen Lexikon gefunden: jahr
Zahl umgesetzt: 1900
Zahl umgesetzt: 74
fußball f"u:s-b,al Sm
weltmeister v"Elt-m,aIs-t6 Sm
Wortteile gefunden: fußballweltmeister:
                        f"u:s-b,al-v,Elt-m,aIs-t6
Wort im kleinen Lexikon gefunden: geworden
...
Wort im kleinen Lexikon gefunden: der
Wort im großen Lexikon gefunden: zeit
```

Interessant ist hierbei, dass sich zusammengesetzte Wörter, findet sich kein exakter Eintrag im Wörterbuch, aus den Teilworten zusammensetzen lassen. Das funktioniert sogar, wenn nur ein Teilwort im Lexikon vorhanden ist. Nur die nicht vorhandenen Teilwörter werden dann nach Regeln umgesetzt.

Das Endergebnis der Graphem-Phonem-Umsetzung ist dann letztlich:

```
{Utterance:begin}{ClauseType:final}
{WordAcc:20}d"OYtS-l,ant
{WordAcc:1}?"Ist+
{WordAcc:1}?Im+
{WordAcc:2}ja:6
{WordAcc:2 0 0 0}nOYntse:nhUnd6t
{WordAcc:0 0 2 0}fi:6Unzi:ptsIC
{WordAcc:2 0 0 0 0}fu:sbalvEltmaIst6
{WordAcc:0 1 0}g@vORd@n+
```

```
{Sentence:end}{Utterance:end}

{Utterance:begin}{ClauseType:final}
{WordAcc:1}vi:+
{WordAcc:2}?alt
{WordAcc:1}v"a:6st+
{WordAcc:1}d"u:+
{WordAcc:1}ts"u:+
{WordAcc:1}de:6+
{WordAcc:2}tsaIt
{Sentence:end}{Utterance:end}
```

Zur Erklärung der Zahlen in z. B. `{WordAcc:2 0}d"OYtS-l,ant`: Die Anzahl der Ziffern zeigt, wie viele Silben das Wort besitzt. Mit dem Zahlenwert wird jeweils die Stärke der Betonung einer Silbe angegeben. Das Wort „Deutschland“ im Beispiel hat zwei Silben, von denen die Erste mit der Stärke „2“, die Zweite dagegen nicht betont wird. Die „+“Zeichen zeigen an, dass es sich bei dem Wort um ein Funktionswort handelt, welche im Allgemeinen kürzer gesprochen werden, da sie sehr oft in der Sprache vorkommen.

2.2.3. Prosodiegenerierung

Im letzten Abschnitt vor der Synthese werden die Dauer und der *f0*-Wert der einzelnen Laute berechnet. Um die Dauer eines Lautes zu ermitteln, wird eine Datei zurate gezogen, welche alle relevanten Daten aller Laute enthält. Ein Eintrag in dieser Datei für das kurze und das lange A etwa sieht folgendermaßen aus:

```
SAMPA  dinh   dmin   prcnt     vokal  lang  kons.  st o f s p
-----------------------------------------------------------
a      100.0  35.0   1.00      1      0     0       0 0 0 0 0
a:     150.0  30.0   1.00      1      1     0       0 0 0 0 0
```

In der Spalte „`dinh`“ wird die inhärente Dauer des Phonems angegeben, in „`dmin`“ die minimale Dauer[8]. „`prcnt`“ ist ein Faktor und wird durch die Anwendung verschiedener Regeln (nähere Informationen in [Kla79]) gewonnen. Steht bei „`vokal`“ eine „`1`“, deutet das naheliegenderweise auf einen Vokal hin, bei „`lang`“ demzufolge auf einen langen Vo-

[8] bei Betonung

kal. „`kons`“ steht für Konsonant, „`st`“ für stimmhaft, „`o`“ für Obstruent[9], „`f`“ für Frikativ, „`s`“ für Sonorant[10] und schließlich „`p`“ für Plosiv.

Aus den Spalten 2 bis 4 wird nach folgender Formel die Lautdauer errechnet[11]:

$$dur := (dur_{inh} - dur_{min}) \times prcnt + dur_{min}$$

Für *TFHTTS* wird das Ergebnis noch mit einem Faktor multipliziert, welcher den lokalen Einfluss widerspiegelt, unter anderem, wie schnell die Sprechgeschwindigkeit eingestellt ist.

Die Protokolldatei von *TFHTTS* nach der Ermittlung der Dauer der einzelnen Laute sieht für das Wort *„Jahr“* folgendermaßen aus:

```
j
Konsonant
noch Folgevokal und Vorgaengerlaut oder
         Nachfolgervokal betont: prcnt = 1.000000
Tempoeinfluss: prcnt = 1.000000
a:
Vokal
Vokal in Einsilb und nicht Fktwort:
                                 prcnt *1.4 = 1.400000
Tempoeinfluss:                   prcnt = 1.400000
6
Konsonant
kein Folgevokal mehr und Vorgaenger- oder
         Nachfolgervokal betont: prcnt = 1.000000
Tempoeinfluss:                   prcnt = 1.000000
```

Dieses Beispiel zeigt, dass lange `a:`-Laute in einsilbigen Worten, welche keine Funktionsworte sind, wie hier das Wort „Jahr“, um 1.4 mal länger gesprochen werden als das durchschnittliche lange `a:`.

Anschließend folgt die Generierung der *f0*-Kurve über der Äußerung. Dabei wird unter anderem der Satzbau, die Wortart und die Betonung innerhalb der Wörter berücksichtig.

[9] eine Gruppe für Plosive, Frikative, Affrikate
[10] Liquide/Laterale, Nasale
[11] wiederum nach [Kla79]

TFHTTS liefert nach erfolgter Generierung folgende Ausgabe:

```
j
j,110
j,.84,0.01 j,.84,0.33 j,.84,0.67

a:
a:,168
a:,.90,0.01 a:,1.00,0.33 a:,1.06,0.67

6
6,60
6,1.08,0.01 6,1.07,0.33 6,1.03,0.67
```

Das lässt sich folgendermaßen interpretieren: In der ersten Zeile wird jeweils der Laut angegeben, in der Zweiten zusätzlich die errechnete Dauer desselben. Die dritte Zeile liefert Stützstellen der *f0*-Werte. Im obigen Beispiel beim Laut `a:` ist die Zeile beispielsweise so zu lesen: 90% der als durchschnittlich festgelegten Grundfrequenz am Anfang des Lautes (1%), 100% bei 33% der Dauer und 107% bei 67% der Dauer des Lautes.

2.2.4. Akustische Synthese

Die Sprachsynthese kennt verschiedene Methoden, um aus der Information über die Art eines Lautes, dessen Dauer und einer Reihe von Grundfrequenzwerten eine akustische Sprachausgabe zu erzeugen. Die beiden bekanntesten und am meisten benutzten sind die sog. Formantsynthese[12] und die Konkatenationssynthese.

Formantsynthese

Die Formantsynthese versucht, eine einfache Wellenform durch Filterung in Sprachsignale umzuwandeln. Die Art der Filterung wird durch viele Regeln umgesetzt, welche eine feine Steuerung von Intonation und Lautdauer erlauben. Prinzipiell ist dieses Vorgehen mit den Vorgängen zu vergleichen, welche bei der Spracherzeugung im Körper eines Menschen stattfinden. Auch hier werden Filter angewandt, um aus einem Basissignal Sprache zu

[12] auch Regelsynthese genannt

erzeugen. Nachteil bei diesem Verfahren ist der sehr künstliche, roboterhafte Klang der erzeugten Stimme.

Konkatenationssynthese

Im Gegensatz dazu die Konkatenationssynthese, welche eine sehr natürliche Ausgabe erzeugt. Hier wird Sprache aufgenommen und in Segmente[13] unterteilt. Diese Segmente werden je nach Bedarf dann wieder in neuer Reihenfolge zusammengesetzt. Problem hierbei ist einerseits, die Schnittstellen akustisch zu verdecken, da sonst unschöne Artefakte entstehen können und andererseits die relativ starre Dauer und Tonhöhe der aufgenommenen Teile. Will man diese verändern, leidet die Natürlichkeit der erzeugten Stimme. Außerdem ist der Speicherbedarf bei der Konkatenationssynthese teilweise um ein Vielfaches höher als bei der Formantsynthese, da die gesamte Datenbank mit der Applikation gespeichert werden muss. Trotzdem arbeiten kommerzielle TTS-Systeme wegen der sehr guten Natürlichkeit der Ausgabe fast ausschließlich damit.

Es gibt verschiedene Spielarten der Konkatenationssynthese. Die *domänenspezifische Synthese* wird verwendet, wenn das Anwendungsgebiet der Applikation genau beschrieben und eingegrenzt werden kann. Hier werden nur die Wörter oder Phrasen, welche für die spätere Sprachausgabe benötigt werden, in die Datenbank aufgenommen. Das ist möglich, weil es eine sehr begrenzte Anzahl von möglichen Äußerungen gibt. Eine denkbare Anwendung ist zum Beispiel eine automatische Ansage der Lottozahlen. Diese Art der Synthese lässt sich schnell und mit vergleichsweise wenig Aufwand realisieren. Und auch die Natürlichkeit der Wiedergabe kann sehr gut sein, da sich *Prosodie* und Betonung in fest umrissenen Aufgabengebieten gut voraussagen lassen.

Die momentan verbreitetste Art der Konkatenationssynthese wird *„Unit Selection Synthese“* genannt. In der (sehr aufwendigen) Erstellung einer Stimme werden Sprachaufnahmen gleichzeitig in Sätze, Phrasen, Wörter, Silben und Laute segmentiert. Diese Segmente werden anschließend in einer Datenbank anhand ihrer lautlichen Eigenschaften wie z. B. ihrem *f0*-Wert indiziert. Bei der Synthese einer Äußerung werden diejenigen Segmente aus der Datenbank geholt und aneinander gefügt, welche am besten passen, ohne dass das Ergebnis digital nachbearbeitet wird. Durch diesen Verzicht verhindert man eine verminderte Qualität der Natürlichkeit des Ergebnisses, kann allerdings auch Parameter wie die *Prosodie* nicht anpassen. Die Qualität dieser Methode reicht an die der *domänenspezifischen*

[13] ganze Wörter, Teile von Wörtern, Laute oder nur Teile von Lauten (sog. Mikrosegmente)

Synthese heran mit dem Vorteil einer, bei entsprechender Datenbankgröße[14], größeren Flexibilität im Anwendungsbereich.

Eine noch weitreichendere Flexibilität lässt sich mit der, unter anderem für diese Arbeit verwendeten, *Diphonsynthese* erreichen. Allerdings muss man Einbußen in der Natürlichkeit der Ausgabe hinnehmen. Dafür lässt sich eine synthetische Stimme mit diesem Verfahrern deutlich schneller und mit relativ geringem Platzbedarf der fertigen Anwendung realisieren.

In der *Diphonsynthese* sammelt man alle Diphone, die in einer Sprache auftreten können. Diphone sind praktisch Lautübergänge zwischen zwei Phonemen. Sie beginnen in der Mitte des ersten Lautes und enden in der Mitte des Zweiten. Abbildung 2.3 zeigt den bairischen Diphon `A-S`. Der Diphon hat seine linke Grenze an der ersten, den Übergang zwischen den Lauten an der zweiten und die rechte Grenze an der dritten gestrichelten Linie.

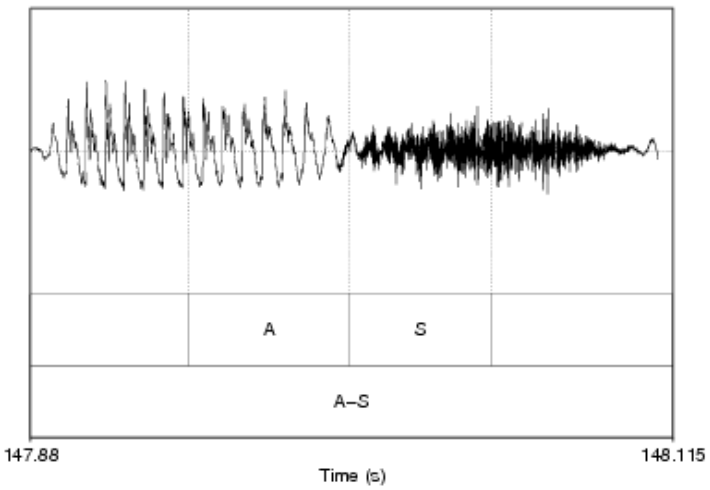


Abbildung 2.3.: Der Diphon `A-S`

Um eine zusammenhängende, möglichst natürlich klingende Äußerung erzeugen zu können, werden die Diphone mithilfe mathematischer Verfahren (wie *PSOLA*, siehe Kap. 2.3.1, Seite 16) mathematischer Funktionen auf die benötigte Länge und den richtigen *f0*-Wert gerechnet und anschließend aneinander gekettet.

[14] welche durchaus Gigabytegröße erreichen kann

2.3. Das Synthesesystem *MBROLA*

Da für diese Arbeit das auf Diphonkonkatenation basierende Synthesesystem *MBROLA*[15] verwendet wird, möchte ich dieses hier im Folgenden kurz beschreiben. *MBROLA* wurde von Dr. Thierry Dutoit am *TCTS Lab* der *Faculté Polytechnique de Mons* in Belgien entwickelt. Es ist für alle gängigen und viele exotischen Plattformen verfügbar und lässt sich genauso wie die benötigten Stimminventare über die Projekthomepage im Internet unter der URL `http://tcts.fpms.ac.be/synthesis/mbrola.html` beziehen. *MBROLA* darf für nichtkommerzielle, nichtmilitärische Zwecke frei verwendet werden. Das *MBROLA*-Projekt hat zum Ziel, die Erforschung der Sprachsynthese, vor allem der Prosodiegenerierung, voranzutreiben und freie Sprachsynthesizer für so viele Sprachen wie möglich bereitzustellen. Das Team unterstützt und berät Entwickler bei der Erzeugung neuer Diphon-Inventare.

Jeder kann ein solches Inventar erstellen, wenn er genügend Zeit und Geduld mitbringt. Die Vorgehensweise dabei ist in Kap. 5 auf Seite 39 beschrieben. Bis zum jetzigen Zeitpunkt wurden bereits 66 von diesen Inventaren geschaffen. Vertreten sind 33 verschiedenen Sprachen, darunter z. B. Isländisch, Litauisch oder das in Indien gesprochene Telugu. Für die deutsche Sprache gibt es 7 verschiedenen Stimminventare. Die Größe der Inventare reicht von ca. 1,8 mb[16] bis fast 55 mb[17]. Das hängt unter anderem davon ab, wie viele Diphone eine Sprache enthält und wie viele davon der Erzeuger implementiert, ob alle *Allophone* implementiert werden usw.

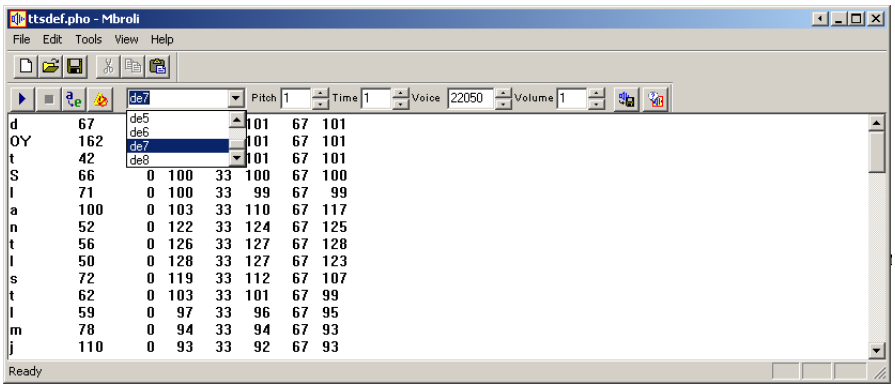


Abbildung 2.4.: Das *MBROLA*-Frontend für Windows, *MBROLI*

[15] ausgesprochen „em brola“, ähnlich dem englischem „umbrella“
[16] die griechische Stimme gr1
[17] die deutsche Stimme de7

In der Windows-Version werden übrigens alle heruntergeladenen Inventare in der Registry eingetragen und lassen sich danach über die Systemsteuerung verwalten.

Für die Windowsplattform existiert außerdem das grafische Frontend *MBROLI* (Abb. 2.4), welches Textdateien mit der Endung `.pho` einliest und die darin enthaltenen Informationen mittels *MBROLA* in Sprache wandelt. Diese Textdateien haben folgendes Datenformat:

```
I    50    0 128   33 127    67 123
s    72    0 119   33 112    67 107
t    62    0 103   33 101    67  99
I    59    0  97   33  96    67  95
m    78    0  94   33  94    67  93
j    110   0  93   33  92    67  93
a:   168   0  99   33 110    67 116
6    60    0 119   33 117    67 113
```

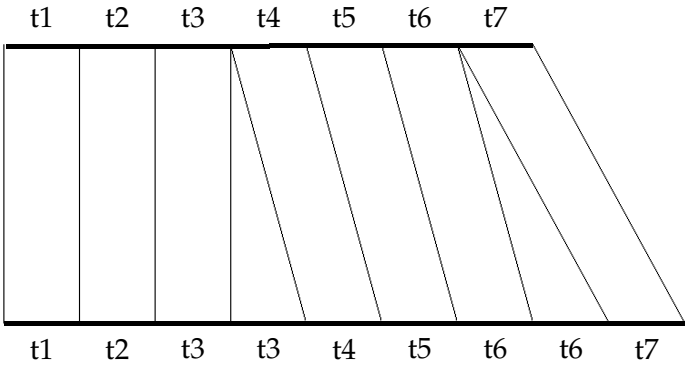
Das Format ist ähnlich dem im Kapitel 2.2.3, Seite 12 Beschriebenen. Die erste Spalte benennt den Laut, die Zweite dessen Dauer. Dann folgen, in diesem Fall wieder drei, es sind jedoch beliebig viele möglich, Stützstellen der *f0*-Kontur. Im Unterschied zum Format von *TFHTTS* muss für die Grundfrequenz hier ein fester Wert (in Hz) angegeben werden.

Neben *MBROLI* gibt es für Windows auch eine Kommandozeilenversion für die *Cygwin*-Umgebung. Entwickler können die *MBROLA*-Engine außerdem in Form einer `.dll` in eigene Applikationen einbinden. Eine genauere Beschreibung der Vorgehensweise dafür findet sich in Kap. 4.1 auf Seite 33.

2.3.1. *PSOLA*

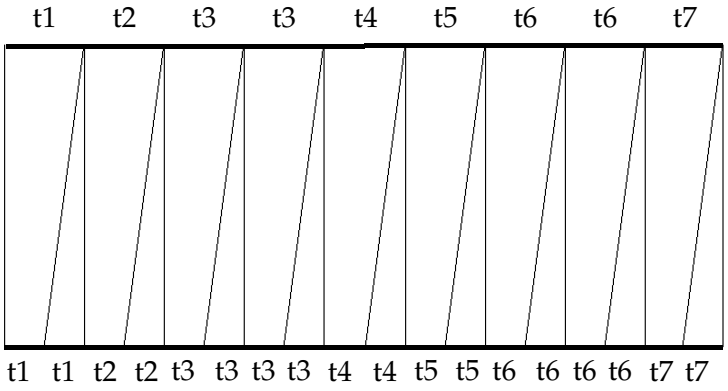
Das Problem beim Zusammensetzen der Diphone ist, dass diese zwangsläufig genau eine Dauer und eine Grundfrequenz besitzen. Es ist jedoch äusserst unwahrscheinlich, dass diese genau denen entsprechen, die gerade für die Synthese benötigt werden. Ändert man aber die Dauer eines Diphons durch einfaches Resampling, ändert sich auch die Grundfrequenz und umgekehrt.

Der Ansatz der von *MBROLA* verwendeten *PSOLA*-Methode, dieses Problem zu lösen, soll im Folgenden kurz erklärt werden.



Die Abbildung oben illustriert das Vorgehen, wenn die Grundfrequenz beibehalten und gleichzeitig die Dauer verlängert werden soll. Es wird so gut es geht versucht, das Signal in periodische Abschnitte zu unterteilen, die anschliessend wieder ineinander überblendet werden. Je nach geforderter Länge wird in den entsprechenden Abständen ein Abschnitt doppelt eingefügt (oder bei Stauchung weggelassen).

Will man jetzt z. B. die Grundfrequenz verdoppeln, wird jedes Segment um die Hälfte gestaucht, verdoppelt und in dieser Reihenfolge dann wieder ineinander gemischt:



Auf diese Weise erhält man ein Ausgangssignal, welches gleichzeitig um die gewünschte Dauer verlängert und bei dem die Grundfrequenz um den gewünschten Faktor erhöht wurde.

3. Der bairische Dialekt

Es ist eigentlich falsch, von „*dem*“ bairischen Dialekt zu sprechen, denn den gibt es selbstverständlich nicht. Die regional gesprochene Sprache unterscheidet sich stark, sobald man sich durch verschiedene Gebiete Bayerns (Abb. 3.1[1]) bewegt. Der bairische Sprachraum

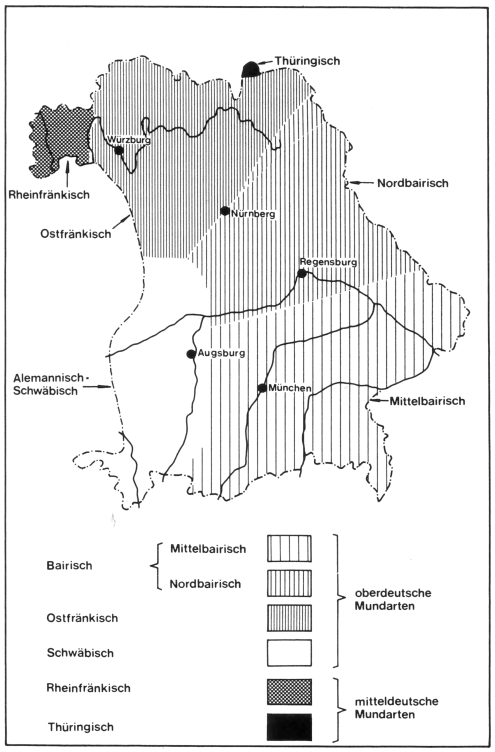


Abbildung 3.1.: Mundarten in Bayern

lässt sich in die Gebiete Nordbayern[2], Mittelbayern[3] und in das völlig außerhalb des bayrischen Landesgebietes liegende Südbayern[4] aufteilen. Der Dialekt, der allgemein als „ty-

[1] entnommen aus [Mer75], Seite 17)
[2] Oberpfalz und das nördliche Niederbayern
[3] Nieder- und Oberbayern, Nieder- und Oberösterreich, Wien und das Burgenland
[4] Tirol, Kärnten und die Steiermark

pisch“ bairisch angesehen wird, wird am ehesten in Ober- und Niederbayern gesprochen. Allerdings ist in den letzten Jahrzehnten eine gewisse „Nivellierung“ der bairischen Mundart zu bemerken, es entsteht eine Art allgemeines Bairisch. Dieses Bairisch nähert sich dem Dialekt, wie er in und um München gesprochen wird an. Außerdem werden immer mehr hochsprachliche und sogar englische Wörter integriert, sozusagen „eingebairt“.

Für diese Arbeit werde ich vor allem die Eigenheiten des oberbayerischen Dialektes beschreiben, mit dem ich aufgewachsen bin und der im Umkreis von Wasserburg am Inn, welches ca. 50 Kilometer südöstlich von München liegt, gesprochen wird.

Für diese Ausprägung der bairischen Mundart habe ich auch das *MBROLA*-Inventar und das Deutsch-Bairische Wörterbuch erstellt, mit welchen das fertige TTS-System arbeitet.

3.1. Die Laute

Als Einstieg ins Bairische werde ich im Folgenden die wichtigsten Laute der Mundart vorstellen und, soweit das überhaupt möglich ist, zu beschreiben versuchen. Dabei nehme ich eine Einteilung in drei Gruppen vor: *Vokale*, *Diphthonge* und *Konsonanten*. Die Phoneme, die sich nicht oder nur gering von ihren hochsprachlichen Entsprechungen unterscheiden, werden der Vollständigkeit halber jeweils am Ende eines jeden Abschnitts aufgelistet. Als Repräsentation verwende ich die *SAMPA*-Symbole, die ich auch für das bairische *MBROLA*-Inventar benutzt werden.

3.1.1. Vokale

Die drei A´s

Der Laut nach dem „R“, der den bairischsprechenden Nichtbayern als erstes enttarnt, ist das A. Genauer gesagt die drei Arten des A´s: a, A und 0. Von Außenstehenden wird der Unterschied fast bzw. gar nicht wahrgenommen. Deswegen versuchen solche Leute meist, das hochdeutsche Normal-a zu verwenden.

Dieses klingt nämlich fast wie die erste Art der bairischen A´s, das *„satte a“*. Das ist lediglich ein wenig dunkler als das Normal-A der Hochsprache[5], wird vielleicht ein klei-

[5] was wiederum die Versuche des Bayern, hochdeutsch zu klingen, zunichte macht

nes bisschen weiter hinten gesprochen als dieses und findet Verwendung in Wörtern wie „k$\underline{a}$sdn“ (Kasten), „v$\underline{a}$R“ (war) und „$\underline{a}$n$\underline{a}$“ (Anna).

Die zweite A-Art ist das sog. *überhelle A*. Das wird vorne gesprochen und klingt ähnlich dem Vokal in den englischen Wörtern *„b$\underline{u}$t“* oder *„m$\underline{u}$st“* (vgl. [Mer75], Seite 9). Es wird oft als bairischer Ersatz für den hochdeutschen a-Umlaut „ä“ benutzt. Beispiele hierfür sind die Wörter „gARdn“ (Gärten) und „SAmA“ (schämen). Auch die Diphthonge „aU“[6] und „OY“[7] werden im Bairischen nicht selten als „A“ ausgesprochen: „b$\underline{A}$m“ (Baum), „r$\underline{A}$mA“ (räumen).

Bei Versuchen, die bairische Sprache durch Ersetzung von Lauten eines hochdeutschen *MBROLA*-Inventars zu „simulieren“ war das überhelle „A“ einer der größeren Problemfälle. Ich habe festgestellt, dass es innerhalb eines Wortes zwar nur wenig Unterschied macht, ob man das „A“ durch ein hochdeutsches „a“, ein „@“(Schwa) oder „6“ ersetzt. Mit ein wenig gutem Willen und in Anbetracht der sowieso vorhandenen Künstlichkeit der synthetisierten Stimme kann man das Ergebnis durchaus als „angebaiert“ durchgehen lassen. Steht das „A“ allerdings am Ende eines Wortes, was häufig vorkommt, denn es ersetzt oft die Graphemkombination „er“ (wie in „aber“), dann lässt sich kein Laut finden, der sich annähernd richtig anhört.

Einige weitere Beispiele für Wörter mit überhellem „A“: „R$\underline{A}$dl“ (Rad), „k$\underline{A}$n$\underline{A}$d$\underline{A}$“ (Kanada) oder, wie oben erwähnt, „ab$\underline{A}$“ (aber).

Interessant ist der Umstand, dass sich nicht immer klar vorhersagen lässt, welcher der beiden a-Laute verwendet werden soll. Bei Namen ist es z. B. möglich, dass es zwei Varianten der Aussprache gibt. So spricht man einen Franz, den man nicht oder nicht sehr gut kennt mit „fR$\underline{a}$nds“ an, wohingegen der beste Freund Franz „fr$\underline{A}$nds“ genannt wird. Das Gleiche passiert dem Max, dem Hans, der Maria und noch einigen anderen.

Das dritte bairische A, das „O“, wird auch „o-a“ genannt, weil es stark in Richtung offenes „o“ geht. Der Laut ist ein hinten bis zentral artikulierter, halb offener Vokal, der mit gerundeten Lippen gesprochen wird. Er klingt ähnlich dem „o“ in den hochsprachlich ausgesprochenen Wörtern „Sonne“, „Gold“. Bairische Beispiele: „s$\underline{O}$l$\underline{O}$d“ (Salat), „h$\underline{O}$dAlUmb“ (Haderlump).

In Abbildung 3.2 werden die drei A-Vokale gegenübergestellt. Anhand der ersten beiden Formanten lässt sich sehr schön erkennen, dass das satte a und das O sehr ähnlich artiku-

[6] *SAMPA*-Notation für „au“
[7] *SAMPA*-Notation für „eu“

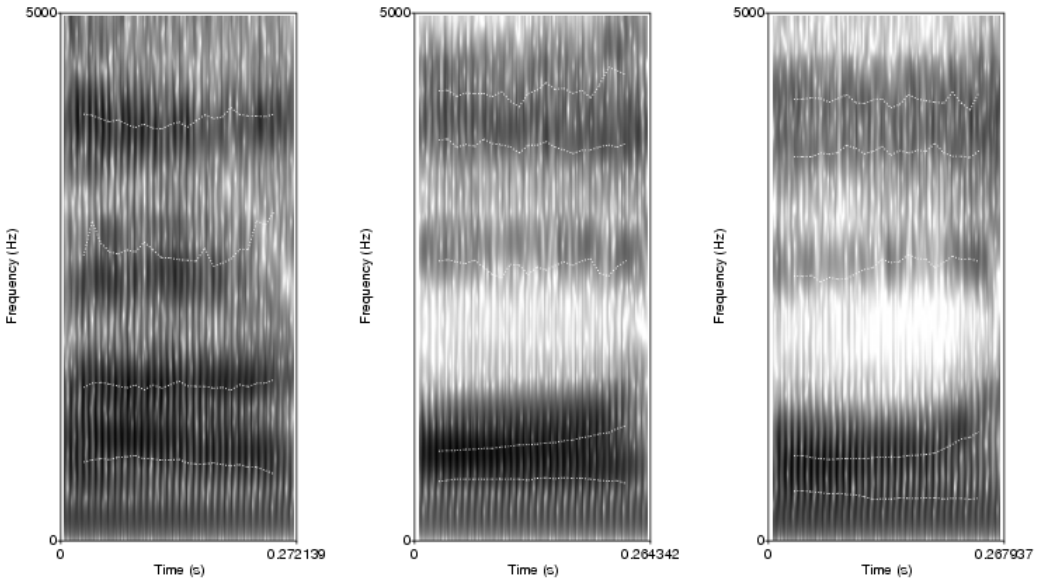


Abbildung 3.2.: Die drei A-Laute des Bairischen

liert werden. Im Gegensatz zum hellen A, welches fast an den Zähnen gebildet wird.

Das „e“

Vom „e“ gibt es zwei Versionen, die sich hauptsächlich anhand ihrer Länge unterscheiden: eben das kurze „e“ und das lange „e:“, wobei das Lange durchschnittlich ca. doppelt so lang gesprochen wird wie das Kurze. Dafür wird das kurze „e“ sehr viel öfter verwendet. Das „e“ ist ein vorne gesprochener, ungerundeter Vokal, der in Bayern immer rein klingt. Es findet sich sehr oft, in seiner unbetonten Form, am Ende eines Wortes, oft als Ersatz für die Endungen „lich“ und „ig“:
„aUsdrYgl$\underline{e}$“ (ausdrücklich), „eAl$\underline{e}$“ (ehrlich), „naIgIAR$\underline{e}$“ (neugierig)

Nasalierung

Es gibt im bairischen Dialekt durchaus nasaliert gesprochene Vokale, doch ist die Nasalierung so fein[8], dass im Rahmen dieser Arbeit darauf verzichtet wurde, diesen Aspekt näher zu untersuchen und nachzubilden.

[8] was allerdings vor einigen Jahrzehnten noch anders gewesen sein mag

Auflistung der verwendeten Vokale

In der folgenden Abbildung werden alle vorhandenen Vokale anhand ihrer ersten beiden Formantfrequenzen in ein Koordinatensystem eingeordnet. Je weiter oben ein Vokal hier zu finden ist, desto weiter vorne befindet sich die Zunge bei der Artikulation desselben. Je weiter der Vokal rechts eingezeichnet ist, desto offener ist Mund.

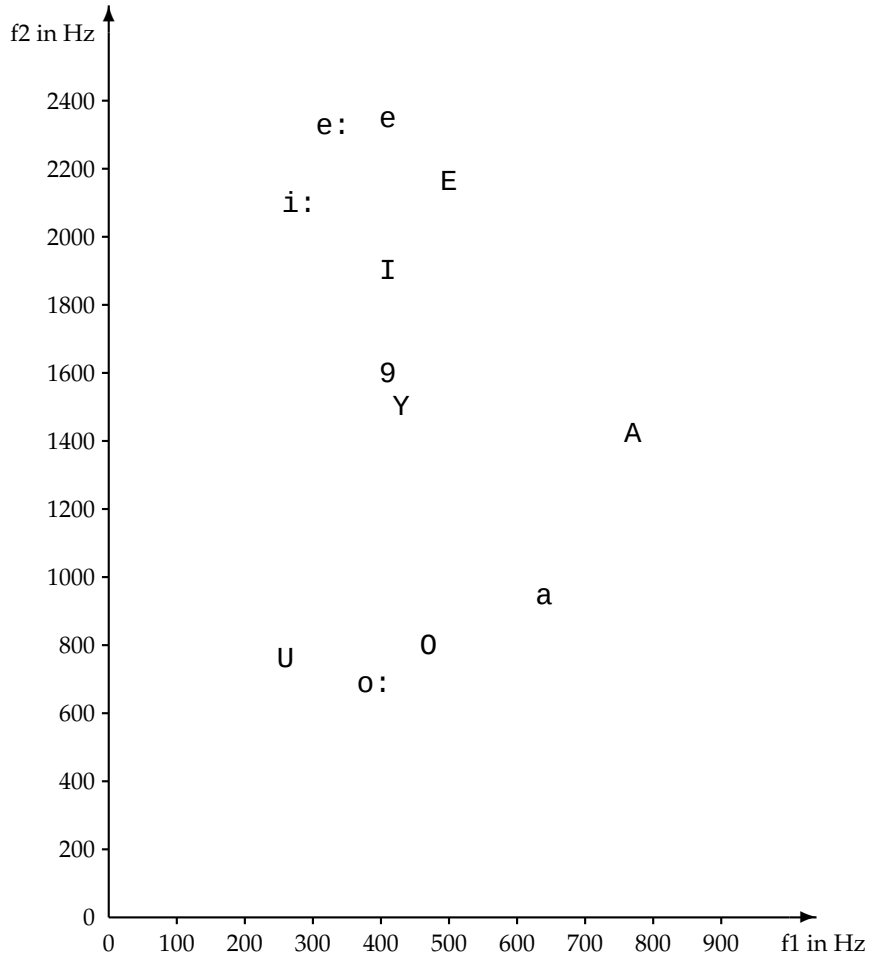


Abbildung 3.3.: Die Lage der Vokale anhand ihrer Formanten

Zusammenfassend werden im Folgenden alle bairischen Vokale in *SAMPA*-Notation aufgeführt, jeweils mit kurzer Beschreibung und Beispiel.

A vorn gesprochenes, helles A, vergleichbar mit Vokal im englischen „but“, z. B. in „RAdl“ (Rad), „tAgsIfaRA“ (Taxifahrer)

a sattes A, z. B. in „ana“ (Anna), „tAgsIfaRA“ (Taxifahrer)

E entspricht dem Ä, wird ausgesprochen wie im Hochdeutschen, z. B. „AdrEs“ (Adresse)

e wie im Hochdeutschem ausgesprochen, z. B. „kenA“ (kennen)

e: langes e, z. B. in „e:s“ (ihr)

I kurzes i, z. B. in „dahIndA“ (dahinter)

i: langes i, z. B. in „i:“ (ich), i:gl (Igel)

o: wie im Hochdeutschen ausgesprochen, z. B. „o:bst“ (Obst)

O eher dumpfer, zwischen dem a und dem offenen o liegender Laut, wird auch als „o-a“ bezeichnet. Zu finden z. B. in „sAlOd“ (Salat) oder „dOg“ (Tag)

9 ö, eher selten im Bairischen, wie in „g9fned“ (geöffnet) oder „k9RbA“[9] (Körper)

U: u, entspricht dem hochdeutschen Laut: „USi:“ (Uschi), „SUs“ (Schuss)

3.1.2. Diphthonge

Ein *Diphthong*, in Bayern auch Zwielaut genannt, ist eine Folge von zwei Vokalen, die durch einen Übergang miteinander verbunden werden.

Die Fülle der *Diphthonge* im Bairischen ist eines der hervorstechendsten Merkmale dieser Mundart. Während in der Hochsprache gerade einmal 3[10] davon gebräuchlich sind[11], kann es in Bayern, je nach Region, bis zu 21[12] Unterscheidbare geben. In und um Wasserburg am Inn habe ich 11 verschiedene Diphthonge ausgemacht, welche ich hier kurz aufzähle und vorstelle:

[9] wurde aber früher und wird auch heute noch manchmal „keAbA“ gesprochen
[10] au (Sampa: aU), ei (Sampa: aI), eu (Sampa: OY)
[11] außerdem gibt es noch 2 Worte mit „ui“: hui und pfui
[12] im Raum Cham, Regen (vgl. [Zeh85], S. 78)

Auflistung der verwendeten Diphthonge

AI lässt sich für Nichtbayern am ehesten als eine Art „äi“ beschreiben, z. B. in „vAId“ (Welt) oder „hAIfde“ (Hälfte)

aU äquivalent zur Hochsprache, z. B. in „maUs“ (Maus)

aI das Hochdeutsche „ei“. „dAfaId“ (verfault), „haIsA“ (Häuser)

eA liegt klanglich irgendwo zwischen „eA“ und „äA“ (hohes A!), Beispiele: „eAnA“ (ihnen), „veAd“ (wird)

IA wird wiederum mit hohem „A“ gesprochen, z. B. in „mIAsn“ (müssen) oder „glavIA“ (Klavier)

OA der wohl bekannteste bairische Diphthong: in OAns (eins, eines), „OAxkAdslSvOAf“ (Eichkätzchenschweif), „dOAg“ (Teig)

OI wird mit dem „o-a“ und kurzem „I“ gesprochen. Kommt u. a. in „kAnOI“ und „vOId“ (Wald) vor.

oI nur sehr geringer Unterschied zum vorhergehenden „OI“, anstelle des „o-a“´s wird eher das normale „o“ gesprochen. Z. B. in „foI“ (voll), „gSdoIn“ (gestohlen)

OU auch einer der typischen bairischen Laute, z. B. in „rOUd“ (rot), „dOUd“ (tot)

UA wiederum mit hohem A gesprochen, „bUASn“ (Burschen) oder „RUA“ (Ruhe)

UI ein eher langes „U“ und kurzes „I“, z. B. „gfUI (Gefühl)“, „SbUI“ (Spiel)

Eine kurze Bemerkung zur Benennung der Diphthonge: Ich habe versucht, alle Laute den *SAMPA*-Konventionen entsprechend zu benennen. Da diese jedoch keine bairischen Diphthonge enthält, wurden die Buchstabenkombinationen gewählt, welche den jeweiligen Einzellauten am nächsten kommen. Mit zwei Ausnahmen: „aI“ und „aU“, welche beide eher mit einem überhellen „A“ gesprochen werden. Da es diese beiden Diphthonge aber auch in der Hochsprache und den für diese erstellten *MBROLA*-Inventare gibt, wo sie jeweils mit eben „aI“ und „aU“ bezeichnet werden, habe ich mich entschlossen, diese Bezeichnung beizubehalten. Im Gegensatz zum normalerweise als „OY“ bezeichneten Zwielaut. Da es hierfür im Bairischen zwei sehr ähnliche Laute gibt, nämlich „OI“ und „oI“, wurde dessen Schreibweise nicht beibehalten, um die Ähnlichkeit vor Augen zu halten.

Das folgende Diagramm zeigt die Lage der Diphone anhand ihrer ersten und zweiten Formanten:

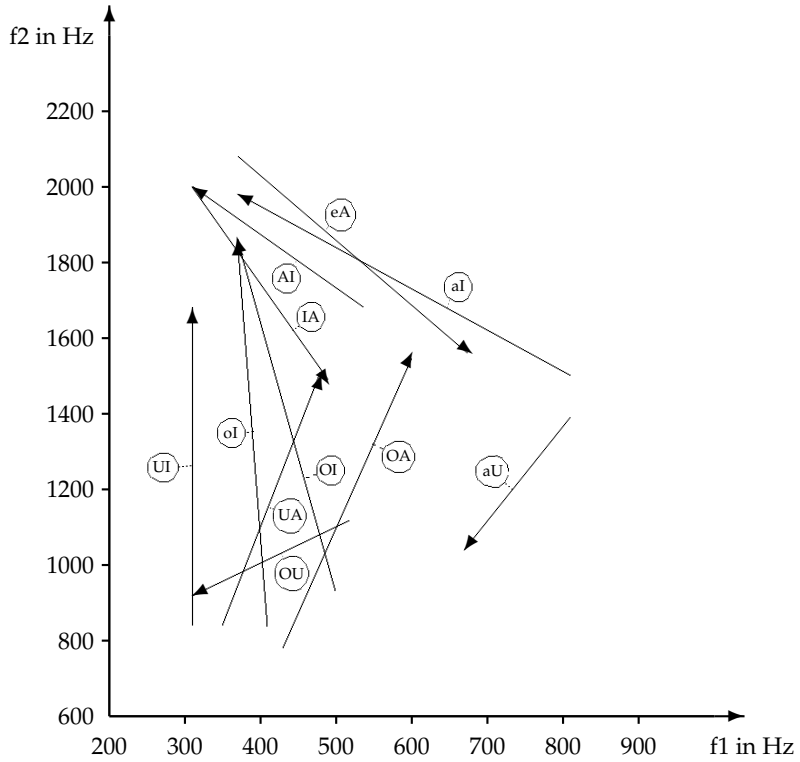


Abbildung 3.4.: Die Lage der Diphthonge

3.1.3. Konsonanten

Der Klang der Konsonanten im Bairischen gleicht im Wesentlichen dem im Hochdeutschen. Es gibt jedoch einige Besonderheiten, welche im Anschluss kurz beschrieben werden.

Das bairische „R“

Der Konsonant, der sich sehr von seinem hochsprachlichen Pendant unterscheidet, ist, wie oben kurz erwähnt, das bairische R. Dieses „r“ wird, im Gegensatz zum Hochdeutschen

mit der Zungenspitze fast an den Zähnen „gerollt“ und ist eines der artikulatorischen Stigmata, die ein Bayer Zeit seines Lebens nicht los wird. Versucht man mit *MBROLA* einen bairischen Satz mit einem hochdeutschen Inventar zu synthetisieren, dann klingen Wörter mit „r“ darin immer, als würde sie ein Amerikaner aussprechen. Die Zunge viel zu weit hinten und ungerollt.

Das „h“

Eine weitere Sonderstellung innerhalb der Konsonanten[13] nimmt der Phonem `h` ein, welcher aus drei Allophonen besteht: „`h`“, „`C`“ und „`x`“. Das bedingt, dass gewisse Diphone nicht existieren, da an einer Position immer nur einer der drei Laute stehen kann. Das „`h`“ beispielsweise kommt nur im Anlaut eines *Morphems* (`haId` – heute) vor oder nach einem Verschlusslaut (`ghaIrAd` – geheiratet). Der Laut „`C`“, der vorne mit der Zunge gebildet wird, tritt nur im In- und Auslaut nach Konsonanten (`manCe` – manche) oder vorderen Vokalen (`fi:C - Tier`) auf. Und der Letzte der Drei, das „`x`“, welches hinten im Rachen artikuliert wird und manchmal sogar „rollt“, findet sich ausschließlich, wie z. B. in „`maxA`“ (machen), im In- und Auslaut nach zentralen und hinteren Vokalen (vgl. [Kuf61], Seite 17).

Das „f/v“

Die Aussprache des „f“ unterscheidet sich nicht von der im Hochdeutschen, wenn man davon absieht, dass das einfach geschriebene „f“ oft, aber nicht immer, länger gesprochen wird.

Das geschriebene „v“ wird in Bayern nahezu immer als „`f`“ gesprochen, einzig mir bekannte Ausnahme ist das Wort „Vase“, welches „`vAsn`“ artikuliert wird.

Auflistung der verwendeten Konsonanten/Konsonantenkombinationen

Hier der Vollständigkeit halber eine Aufzählung aller Konsonanten und Konsonantenkombinationen, jeweils mit Beispiel, welche im *MBROLA*-Inventar verwendet werden:

[13] nicht nur im Bairischen

SAMPA	Beispiel	SAMPA	Beispiel
b	bo:lIdsaI (Polizei)	**l**	lambm (Lampe)
pf	pfeAdl (Pferd)	**m**	hamA (Hammer)
C	fi:C (Vieh)	**n**	naxAd (nachher)
x	RUAx (geiziger Mensch)	**N**	aNgsd (Angst)
d	dOC(Dach)	**R**	RUmben (rumpeln)
ds	blOds (Platz)	**s**	mas (Maß)
f	fAgesn (vergessen)	**S**	bolIdIS (politisch)
g	agA (Acker)	**Z**	ZUId („die Schuld“, „schuld“ sein)
h	haUs (Haus)	**t**	tUAm (Turm)
j	javoI (jawohl)	**v**	vOAnA (weinen)
k	kUA (Kuh)		

3.2. Bairische Eigenarten

3.2.1. Entrundung

Im bairischen Dialekt sind die beiden Vokale „9“ (ö) und „Y“ (ü) eigentlich nicht vorgesehen. Durch eine Eigenart, die sog. „Entrundung“, werden bei Lauten, welche an sich mit gerundeten Lippen gesprochen werden, die Lippen breit gezogen. So entsteht aus einem „ö“ ein „e“ oder im Extremfall, ein „ä“. Aus „schön“ wird „Se:“ und aus „böse“ „bEs“. Das „ü“ mutiert zum „i“ wie in „glIg“ (Glück). Auch der im Hochdeutschen vorkommende Diphthong „OY“ (eu) kann entrundet werden. Er wird zu einem „ei“ (aI). Ein Beispiel hierfür: Aus „freundlichen Leuten“ werden „fRuaIndliCe laId“.

Es lässt sich allerdings beobachten, dass die Entrundung nicht mehr so konsequent betrieben wird wie früher. Vor allem in städtischen Gebieten werden die „ö“´s und „ü“´s auch vermehrt als solche gesprochen. Auch bei Wörtern, die dem bairischen Wortschatz nachträglich hinzugefügt wurden, sei es durch neue technische Entwicklungen oder fremdsprachlichen Einfluss, wird die Entrundung nicht mehr, zumindest nicht immer angewendet. Der „Föhn“ wurde nicht zum fe:n und der „Euro“ nicht zum aIRo:[14].

[14] wobei manche ältere Menschen manchmal diese Aussprache wählen

3.2.2. Umlaute

Nicht alle Umlaute werden allerdings entrundet, nämlich, wenn im hochdeutschen Wort ein „ck“, ein „pf“ oder „tz“ dem Umlaut „ü“ folgt. Dieser Umstand verhindert oft eine Entrundung und lässt deshalb aus der „Brücke“ nur eine „brUgn“ oder aus „nützen“ „nUdsn“ werden (vgl. [Mer75], S. 16).

Auch das „ä“ kann nicht entrundet werden, da es ja nicht rund gesprochen wird. Für die Ersetzung dieses Vokals lassen sich keine allgemein gültigen Regeln definieren.

Es wird oft zu einem überhellen „A“, wie man an „gSAfdIg“ (geschäftig) oder „jAgA“ (Jäger) sehen kann. Manchmal mutiert es zu einem „e“, wie in „kembfA“ (kämpfen). Außerdem kann es auch passieren, oft bei Wörtern im Plural, dass das a erst gar nicht umgelautet wird und als Sattes ausgesprochen wird: manA[15] (Männer).

Allermeistens aber wird das „ä“ in der heutigen Zeit einfach unverändert als „E“ gesprochen. Oder, falls ein „r“ folgt, wird aus „är“ der Diphthong „eA“.

3.2.3. Konsonantenschwächung

Ein wesentlicher Unterschied zwischen der Hochsprache und der Mundart besteht in der sog. Konsonantenschwächung. Diese bezeichnet die annähernde Aufhebung des Unterschieds zwischen *„b“* und *„p“*, *„d“* und *„t“* und teilweise[16] auch zwischen *„k“* und *„g“*. In manchen Regionen werden sogar noch *„b“* und seltener das *„g“* erweicht. Das *„b“* wird zu *„w“* wie in „kI:ve“ (Kübel) und das *„g“* zu *„ch“* (dsaIC – Zeug). Interessant ist in diesem Zusammenhang, dass der Laut „p“ in der bairischen Sprache außer im Doppellaut „pf“ am Anfang eines Wortes überhaupt nicht vorkommt. Genauso wie der Laut *„t“*, welcher sich auch nur am Beginn eines Wortes findet. Oder in Wörtern, die entweder aus der Hochsprache übernommen wurden oder durch fremdsprachlichen Einfluss dem Wortschatz zugefügt wurden (z. B. Status, Ethos).

[15] wird heutzutage aber auch manchmal „menA“ gesprochen
[16] vor allem, wenn sie vor Konsonanten stehen

3.3. Bairische Grammatik

Dieser Abschnitt stellt einige wenige ausgewählte Regeln der bairischen Grammatik vor, um die Problematik beim Übersetzen in die Mundart anzudeuten. Für Interessierte findet sich viel weiterführendes Material in [Mer75] und [Zeh85].

3.3.1. Das Substantiv

3.3.2. Die Artikel

Die Aussprache der bestimmten Artikel im Bairischen verändert sich, je nach Betonung desselben. Die folgende Tabelle[17] stellt die verschiedenen Formen gegenüber:

	betont	unbetont	nach Präposition
mask.			
Nom.	deA	dA	- -
Dat.	den/dem	An/Am	n/m
Akk.	den	An	n
neutr.			
Nom.	de:s	As/s	- -
Dat.	dendem	An/Am	n/m
Akk.	de:s	As/s	n
fem.			
Nom.	de:	d	d
Dat.	deARA	dA	dA
Akk.	deÖ	d	d
Plur.			
Nom.	de:	d	d
Dat.	dene	de:	de:
Akk.	de:	d	d

So wird beispielsweise „Die/diese Kuh kaufe ich.“ „de: kUA kAf i:“ gesprochen, im Gegensatz zu „Die Kuh kalbt.“, „d kUA koIwAd“. Ganz richtig in Bezug auf den genauen Klang der Aussprache ist obige Tabelle übrigens auch nicht. Es gibt nämlich manche

[17] entnommen aus [Zeh85], Seite 111)

Konsonanten, vor denen man den Artikel „die“ nicht auf „d“ verkürzt, wie bei „d kUA“, sondern auf ein „b“. „bfegl sINAn.“ – „Die Vögel singen.“. Das „d“ assimiliert sich an den folgenden Konsonanten und gleicht sich ihm an (vgl. [Mer75], Seite 86 ff.)

Ein weiterer Unterschied zeigt sich beim Verwenden des unbestimmten Artikels *„ein“*:

- Wird das „ein“ als unbestimmter Artikel verwandt, spricht es der Bayer „A“:
  „gIb mIA *A* bUAx.“ – Gib mir (irgend-)ein Buch.
- Verwendet man es als Numeral, ist die Aussprache „OA“:
  „gIb mIA *OA* bUAx.“ – Gib mir (genau) ein Buch.

3.3.3. Das Wortgeschlecht

Auch das Wortgeschlecht unterscheidet sich in einigen (und nicht wenigen) Fällen von dem der Hochsprache. Es lässt sich keine Regel aufstellen, wann das Geschlecht gleich bleibt, oder wann es sich ändert. Eines der bekannteren Beispiele ist wohl *„der* Butter“ in Bayern. Es lassen sich aber noch viele Substantive aufzählen, die sich im Geschlecht unterscheiden: *„der* Radio“, *„das* Teller“, *„der* Socke“, *„die* Spachtel“, um nur einige zu nennen. Wollte man diese Art Unterschied zum Schriftdeutschen bei einer automatischen Übersetzung abfangen, müsste eine komplette Liste aller Unterschiede aufgestellt werden und das *TTS-System* müsste in die Lage versetzt werden, zu erkennen, wann es angebracht ist, den Artikel zu ändern. Danach muss noch die in Kap. 3.3.2 beschriebene eventuelle Verkürzung desselben berücksichtigt werden.

3.3.4. Singular und Plural

In der Hochsprache enden viele Substantive auf ein unbetontes „e“. Das Bairische läßt dieses entweder einfach weg (kads – Katze, Sdras – Strasse) oder ersetzt es durch ein „n“ (beheAdn – Behörde, brUIn – Brille). Lautlich kann aus dem „n“ aber in der Folge auch ein „m“ (lambm – Lampe), ein „N“ (so:RN – Sorge) oder sogar ein „A“ (SdOAnA – Steine) werden.

Wird ein „n“ angehängt, so ist diese Form auch gleichzeitig die Aussprache nach der Pluralbildung des Substantives. Es gibt auch den Fall, dass, während in der Hochsprache für den Plural ein „e“ am Ende angefügt wird, dieses im Bairischen wieder wegfällt (BeAg –

Berg/Berge, `bedRi:b` – Betrieb/Betriebe). Manchmal wird auch, wie im Hochdeutschen, der Stammvokal mit umgelautet, wie in „`asd/Esd`“ (Ast/Äste).

Während im Hochdeutschen der Plural bei Wörtern, die mit „en“, „chen“, „lein“ enden, endungslos bleibt (die Eisen, die Mädchen) und solche mit „er“ und „el“ unterschiedlich behandelt werden (die Kartoffeln, die Deckel), hängt das Bairische in solchen Fällen immer ein „n“ an (`kaRdo:fe/kaRdo:fen`, `degl/degln`).

Es gibt noch einige weitere Regeln zur Bildung des Plurals. Diese hier auszuführen würde aber doch zu weit führen.

3.3.5. Weitere Unterschiede

Im Folgenden eine kurze, unvollständige Aufzählung weiterer wichtiger Unterschiede zwischen der bairischen und der hochsprachlichen Grammatik:

- Das Bairische kennt keinen Genitiv. Aus: „Das Fahrrad der Schwester.“ wird „`dA SvesdA saI`[18] `RAdl.`“ oder „s RAdl vo: dA SvesdA.“
- Die Vergangenheitsform Präteritum existiert nicht, außer bei Hilfsverben (ich war, ich wollte). Sie wird immer durch das Perfekt ersetzt, also wird aus „Ich rechnete die Aufgabe.“ „`i: hOb de aUfgabm gReCnEd.`“.
- Es gibt keinen Konjunktiv Präsens und keinen Konjunktiv Perfekt in der Mundart. Er wird durch den Indikativ ersetzt: „Sie sagt, er habe gewonnen.“ wird zu „`si: sOgd, eA hOd gvUnA.`“.
- Im Gegensatz zur Hochsprache bedeutet mehrfache Verneinung immer eine Verstärkung der Verneinung. „`eA hOd mIA nIA kOA gAId ned gem.`“ (Er hat mir nie kein Geld nicht gegeben.) heißt sinngemäß: „Er hat mir wirklich nie Geld gegeben, ich schwörs.“[19]
- Es stehen fast ausnahmslos Artikel vor Personennamen, Verwandtschafts- und Berufsbezeichnungen (vgl. [Zeh85], Seite 115). Nie findet man etwas wie „Franz kommt gleich“. Immer „`dA frands kImd glaI`“.
- usw.

[18] man beachte den Verzicht auf die Geschlechtsunterscheidung (`saI` = ihr)

[19] Ein schöner Satz auch die Vierfachverneinung: „baI uns hOd no: nIA kOAnA KOAn hUngA ned lAIn mIAsn!“ (Bei uns hat noch nie keiner keinen Hunger nicht leiden müssen!“)

Teil II.

Praktischer Teil

4. Anbindung von *MBROLA* an *TFHTTS*

4.1. Die *MBROLA*-Bibliothek „`mbrplay.dll`“

Thierry Dutoit, der Autor von *MBROLA* stellt neben dem Frontend *MBROLI* auch zwei *dll´s* bereit, um *MBROLA* in eigene Programme einzubinden. Die Erste davon, `mbrola.dll`, ermöglicht die Benutzung von *MBROLA* anhand relativ grundlegender Funktionsaufrufe. So ist zum Beispiel die Vorgehensweise, um eine `.pho`-Datei in eine `.raw`-Datei umzuwandeln Folgende:

1. `mbrola.dll` laden:
   `load_MBR();`
2. das gewünschte Inventar laden:
   `init_MBR(`*pfad_zum_inventar*`);`
3. zeilenweises einlesen des `.pho`-Files
4. schreiben des eingelesenen Inhaltes in den Buffer des Synthesizers:
   `write_MBR(`*eingelesene_zeile*`);`
5. durchführen der Synthese:
   `read_MBR(`*zeiger_zum_ausgabearray*`,` *laenge_des_arrays*`);`
6. zeilenweises schreiben des Syntheseergebnisses in die `.raw`-Datei
7. `mbrola.dll` mitteilen, dass man fertig ist, damit diese den Buffer flushen kann:
   `flush_MBR();`
8. schließen und entladen der `mbrola.dll`:
   `close_MBR();`
   `unload_MBR();`

Bei der Zweiten *.dll*, `mbrplay.dll` ist die Prozedur für die gleiche Aufgabe ungleich kürzer:

1. laden des gewünschten Inventars:
   `MBR_SetDatabase(`*pfad_zum_inventar*`);`
2. synthetisieren des `.pho`-Files und speichern des Ergebnisses:
   `MBR_Play(.pho-`*File*`,MBR_BYFILE|MBROUT_RAW|MBR_WAIT,.raw-`*File*`,NULL);`
3. entladen von *MBROLA*:
   `MBR_MBRUnload();`

Bei beiden Methoden sollte vor bzw. nach jedem Befehl während der Ausführung auf Fehler kontrolliert werden. Dies ist entweder mit dem Rückgabewert der einzelnen Funktionen möglich oder in anderen Fällen mit einer Zeile Code der Art:

```
if (MBR_LastError(NULL,0)<0){ code für Fehlerfall... }
```

Der Übersicht halber wurden diese Zwischenschritte weggelassen. Eine genaue Beschreibung und Auflistung aller verfügbaren Funktionen beider *dll´s* findet sich in der Dokumentation der Windowsdistribution von *MBROLA*.

4.2. Die Anpassung von *TFHTTS*

TFHTTS besteht im Wesentlichen aus der Datei `tfhtts.dll`. Diese liefert die benötigte Funktionalität von der Vorverarbeitung bis zur fertigen Sprachausgabe. Zum Testen / Benutzen des Systems existiert das Frontend `eval_tts.exe` (Abb. 4.1).

Beide Applikationen wurden ca. 1997 mit der Entwicklungsumgebung *Visual Studio 6* von *Microsoft* in der Programmiersprache *C++* geschrieben. Da für diese Arbeit lediglich der Teil von *TFHTTS* ersetzt werden musste, der die akustische Synthese erledigt, waren keine allzu großen Eingriffe in beide Programme notwendig.

4.2.1. Die Oberfläche `eval_tts.exe`

Wie in Abb. 4.1 zu sehen ist, bietet *eval_tts* die Möglichkeit, die Geschwindigkeit und die Grundfrequenz der Ausgabe zu verändern. Außerdem lässt sich angeben, welche Sprache der zu synthetisierende Text hat.

Letzteres ist nach der Modifizierung nicht mehr notwendig, weil davon ausgegangen werden kann, dass die Sprache immer Deutsch ist. Deswegen kann dieses Eingabefeld dazu

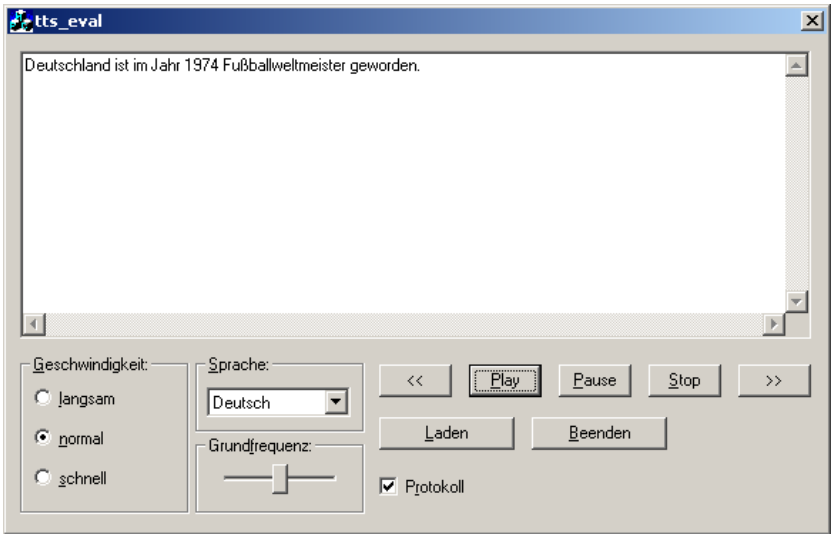


Abbildung 4.1.: Die grafische Oberfläche zu *TFHTTS*, `eval_tts.exe`

verwendet werden, alle auf dem System installierten *MBROLA*-Inventare anzuzeigen und den Benutzer eines auswählen zu lassen.

Um sich eine Liste aller auf dem System installierten Stimmdatenbanken ausgeben zu lassen, hält `mbrplay.dll` die Funktion

```
MBR_RegEnumDatabaseCallback(dbCallback, (DWORD)&m_cLanguage);
```

bereit. Wie der Name schon andeutet, ist diese Funktion eine Callback-Funktion, das heißt, sie bekommt als Parameter den Namen einer Funktion übergeben, im vorliegenden Fall ist der Name „`dbCallback`“. Der zweite Parameter, „`(DWORD)&m_cLanguage`“ ist der Name eines auf den Datentyp `DWORD` gecasteten, dereferenzierten Zeigers. Dieser zeigt in unserem Fall auf die Eigenschaft der Auswahlbox, welche zur Laufzeit dann die Auswahlmöglichkeiten der Komponente bereitstellt, die eine direkte Änderung des zu benutzenden Inventars ermöglicht.

Die Funktion geht der Reihe nach alle *MBROLA*-Inventare des Systemes durch. Für jedes Gefundene wird die übergebene Funktion `dbCallback` ausgeführt. Der erste Übergabeparameter der Funktion enthält dabei das gerade gefundene Inventar, der zweite ist ein Container für die Daten, welche man aus der Funktion zurückgeben möchte. In unserem Fall ist das ein String, der alle Namen aller im System vorhandenen *MBROLA*-Inventare enthält.

Hier die Funktion:

```
BOOL CTts_evalDlg::dbCallback
      (LPCTSTR lpszDatabase, DWORD dwUserData)
{
  char label[_MAX_PATH];

  MBR_RegGetDatabaseLabel(lpszDatabase,label,_MAX_PATH);

  (*(CComboBox*)dwUserData).AddString(label);
  return TRUE;
}
```

Zum Ablauf in der Callback-Funktion: Erst wird eine Variable (`label`) definiert, welche den Namen der Stimmdatenbank durch den Befehl `MBR_RegGetDatabaseLabel` zugewiesen bekommt. Danach wird diese Variable an den (beim ersten Mal noch leeren) Container angehängt.

Durch aufrufen der obigen Callback-Funktion `MBR_RegEnumDatabaseCallback` beim Initialisieren der grafischen Oberfläche (in der von *MS Visual Studio* standardmäßig erzeugten Methode `BOOL CTts_evalDlg::OnInitDialog()`) lässt sich die Auswahlbox mit der Liste aller vorhandenen Inventare füllen.

Etwaige Änderungen der Grundfrequenz durch den Benutzer könnten zwar durch den Aufruf der Methode `MBR_SetPitchRatio(SliderVals[m_cGFSlider.GetPos()])` abgefangen und *MBROLA* direkt mitgeteilt werden. Da die Werte aber sowieso der Berechnungseinheit zur Verfügung gestellt werden, wurde hier keine Anpassung vorgenommen. Es wurde lediglich noch ein Anzeige der aktuellen Grundfrequenz in die GUI eingebaut, um dem Benutzer immer die aktuelle Grundfrequenz anzeigen zu können.

Um *MBROLA* die gewünschte Sprechgeschwindigkeit mitzuteilen, muss am Programm genauso wie oben nichts geändert werden, da die Prosodiegenerierung von *TFHTTS* diesbezügliche Änderungen des Benutzers sowieso berücksichtigt. Wäre dem nicht so, wäre es aber kein Problem, *MBROLA* vor dem Abspielen einen bestimmten Tempofaktor vorzugeben. Die Befehlssyntax hierzu lautet `MBR_SetDurationRatio(float fDuration);`.

4.2.2. Die Bibliothek `tfhtts.dll`

Ein wenig mehr Aufwand ist für die Anpassung der Bibliothek `tfhtts.dll` notwendig. Der Kern von `tfhtts.dll` ist die Quellcodedatei „`Ttsmain.cpp`“. Dort findet sich die Methode `DWORD SynthesizeText(LPVOID Daten)`, in welcher der Reihe nach alle notwendigen Schritte von der Vorverarbeitung bis zur Prosodiegenerierung (siehe Kap. 2.2, Seite 6 abgearbeitet werden.

Der erste Eingriff erfolgt nach der Berechnung der Grundfrequenzkontur über eine Phrase. Diese wird in der Methode `int intona( char *filestring )` vorbereitet, welche in der Quellcodedatei `into_fuj.cpp` definiert ist. Durchgeführt wird die Berechnung mittels der Methode `void inton_fujisaki(...)` aus der Datei `fuji_fnc.cpp`. Wenn man das Ergebnis der Methode `inton_fujisaki` erhält, liegt es im in Kap. 2.2.3 auf Seite 12 beschriebenen Format vor. Zur Erinnerung:

```
j
j,110
j,.84,0.01 j,.84,0.33 j,.84,0.67
```

Zielformat ist das von *MBROLA* Verwendete (siehe Kapitel 2.3, Seite 16) mit festem Grundfrequenzwert:

```
j    110   0  93   33  92    67  93
```

Für die dazu notwendige Umwandlung vom einen ins andere Format wurde eine Klasse `CMbrolaTranslation` in der gleichnamigen `.cpp`-Quellcodedatei erstellt, welche lediglich die Funktion

```
short CMbrolaTranslation::dat_aus_pho(...)
```

enthält. Basis derselben ist die Funktion `short dat_aus(...)`[1], welche den Erfordernissen *MBROLA*s entsprechend umgebaut wurde (die Klasse `CMbrolaTranslation` findet sich im Anhang B).

Außerdem wurde noch sichergestellt, dass die ursprüngliche hochdeutsche Version von *TFHTTS* parallel mit der bairischen weiterbenutzt werden kann. Dafür ist die in der Datei `globals.cpp` definierte globale Variable „`char gLanguageString[4]`“ wichtig. Diese enthält das Kürzel der zu synthetisierenden Sprache. Das Kürzel ist auch in den Namen aller Inventar- und GPU-Dateien einer Sprache (in der Form: `Lex_ger.dat`) vorhanden.

[1] in der Datei `into_fnc.cpp`

Der entsprechende Wert wird der Variablen in der Funktion `LangID2LangExt`, welche man in der Quelldatei `Language.cpp` finden kann, zugewiesen. Um einzustellen, ob die hochdeutsche oder die bairische Variante benützt werden soll, wird geprüft, ob aktuell das *MBROLA*-Inventar mit dem Namen *„de8“* geladen ist. Ist dies der Fall, wird die bairische Version der Dateien in den Speicher geladen. Im Folgenden der (leicht gekürzte) Quellcode der Funktion `LangID2LangExt`:

```
void LangID2LangExt(LANGID dwLangID,char *pszLangExt)
{
char label[4];

// name der mbrola-db abfragen und entspr. sprache setzen
MBR_GetDatabase(label, _MAX_PATH);

if ( !strcmp(label, "de8") ) { strcpy(pszLangExt, "bai"); return;}
else if ( !strncmp(label, "de", 2) ){ strcpy(pszLangExt, "ger");
return;}
else if ( !strncmp(label, "fr", 2) ){ strcpy(pszLangExt, "fr" );
return;}
else if ( !strncmp(label, "uk", 2) ){ strcpy(pszLangExt, "uk" );
return;}
else if ( !strncmp(label, "us", 2) ){ strcpy(pszLangExt, "us" );
return;}
else if ( !strncmp(label, "it", 2) ){ strcpy(pszLangExt, "it" );
return;}
else { strcpy(pszLangExt,""   );
TRACE("LangID2LangExt: Language not Supported\n");
return;
}}
```

„de8“ ist der Name des bis auf weiteres einzigen bairischen Inventars. Sollte es tatsächlich irgendwann ein Zweites geben, müßte man dessen Namen hier nachtragen.

5. Erstellen des *MBROLA*-Inventars

5.1. Vorbereitung der Phoneme und Diphone

Grundvoraussetzung für die Erstellung eines *MBROLA*-Inventars ist eine vollständige Liste aller zu verwendenden *Phoneme* und möglicherweise *Allophone* davon. Die vollständige Auflistung der von mir verwendeten Laute findet sich in Kapitel 3.1 ab Seite 19.

Danach ist eine vollständige Aufstellung aller in der Zielsprache vorkommenden *Diphone* anzufertigen. Das beinhaltet auch *Diphone*, die nie innerhalb eines Wortes auftreten, da es durchaus möglich ist, dass sie erst durch das zusammenhängende Sprechen von zwei Wörtern entstehen. So gibt es z. B. kein bairisches Wort, in welchem die beiden Diphthonge `aU` und `eA` nacheinander vorkommen. In der Wortfolge

`SaUeAmnedO` (Schau ihn nicht an,...)

ist diese Kombination aber trotzdem enthalten. Praktisch heißt das, dass fast jedes Phonem und Allophon mit jedem anderen Phonem/Allophon kombiniert werden kann, da alle hin und wieder am Wortanfang bzw. Wortende vorkommen. Ausnahmen sind hier die Allophone „`C`“ und „`x`“, sowie „`N`“, welche nie am Wortanfang stehen. Deswegen ist es nicht nötig, bestimmte Diphone, die einen der betreffenden Laute enthalten, in das Inventar aufzunehmen. Beispiele hierfür: „`f-x`“ bzw. „`f-C`“, „`j-N`“... Für die Allophone „`C`“, „`h`“ und „`x`“ gilt außerdem, dass in Verbindung mit Vokalen nur immer einer der drei Laute verwendet wird (siehe Kap. 3.1.3, Seite 26), weswegen für den Anderen kein Diphon mit dem gleichen Partner aufgenommen werden muss.

Eine weitere Ausnahme betrifft wiederum das „`h`“. Das wird nie am Ende eines Wortes gesprochen und außerdem nur von Vokalen bzw. Diphthongen gefolgt. Deswegen kommen Kombinationen, in denen das „`h`“ links und ein Konsonant rechts steht, nie vor und brauchen auch für das Inventar nicht berücksichtigt zu werden.

Die beiden Vokale „`Y`“ und „`9`“ würden im Bairischen im Grunde nie am Ende eines Wortes stehen, doch ist es durchaus möglich, dass Fremdwörter in einem Text auftauchen, bei

denen das vorkommt (z. B. Queue oder Camus[1]). Aus diesem Grund wurden beide Laute in allen Kombinationen berücksichtigt.

Nach dem Erstellen der Liste müssen Trägerworte, welche die Diphone enthalten, gefunden werden. Die Diphone sollten an einer annähernd nichtzentralen, nichtbetonten Stelle innerhalb des Wortes stehen, um einen möglichst neutralen Klang zu erhalten. Würde nämlich ein betonter Laut an einer unbetonten Stelle eines Wortes resynthetisiert, könnte es Probleme mit der Natürlichkeit des Klanges geben.

Für die Auswahl geeigneter Trägerworte schlagen die Macher von *MBROLA* drei verschiedene Arten vor[2]:

Die erste Methode besteht darin, für jeden verwendeten Laut je zwei Trägerworte zu finden. Eines, das mit dem Laut beginnt und eines, das mit ihm endet. Hat man für jedes Phonem zwei solcher Worte gefunden, werden diese je nach Zusammensetzung des Diphons kombiniert.
Soll der Diphone `OI-b` aufgenommen werden, verbindet man z. B. die bairischen Wörter `kAnOI` (Kanal) und `bo:lidsaI` (Polizei) zur in der Praxis nie vorkommenden

`kAnOIbo:lidsaI`.

Später wird dann der Diphon `IA-b` mit dem Wortpaar `glAvIA` (Klavier) und wiederum `bo:lidsaI` gebildet:

`glAvIAbo:li:dsaI`

usw.

Die zweite Methode besteht im Erfinden von Fantasiewörtern, aus denen die Diphone extrahiert werden. Aus dem Wort `glo:ri:fUAgEn` ließe sich die Lautkombination `f-UA` gewinnen. Vorteil hierbei ist, dass der Sprecher nicht so leicht in Versuchung gerät, Silben zu betonen, da er auf keine Erfahrungen in der Aussprache des Wortes zurückgreifen kann.

Die letzte Methode kann in Frage kommen, wenn man im Besitz eines phonetischen Wörterbuches der Zielsprache ist. Man kann dann ein Skript benutzen, um sich eine Auswahl von fraglichen Trägerworten automatisch aussuchen zu lassen. Letztlich muss man aber auch hier noch eine manuelle Endauswahl treffen, da manche linke und rechte Kontexte des Diphons besser geeignet sind als andere. Die Lautfolge `SdAIUNsfElA` (Stellungsfehler) z. B. eignet sich nicht besonders, um den Diphon (U-N) zu extrahieren. Erstens dürfte

[1] wobei die Wahrscheinlichkeit, dass der Bayer diese korrekt ausspricht, nicht allzu hoch ist
[2] für eine detailliertere Anleitung siehe [tm00]

wohl, da ja eigentlich drei Vokale nacheinander stehen (A, I, U), die vordere Lautgrenze des „U"´s schwierig zu finden sein, selbst wenn das „I" und das „U" im Vokalviereck relativ weit auseinander liegen. Außerdem ist zu bezweifeln, dass das „U" in dieser Kombination sehr stabil und exakt ausgesprochen wird.

Abschließend sollte man versuchen, einen Trägersatz zu finden, in das sich das Trägerwort möglichst natürlich einfügt. Der Satz sollte ohne Schwierigkeiten monoton zu sprechen sein und das Trägerwort sollte bevorzugt an unbetonter Stelle stehen. Das Wichtigste ist, dass sich das Zielwort nicht am Anfang des Satzes befindet, da es bei jedem Sprecher eine gewisse Zeit braucht, bis sich seine Stimme stabilisiert hat.

Für das bairische Inventar habe ich hauptsächlich die erstgenannte Methode zum Produzieren von Trägen verwendet. Für einige häufig auftretende Diphone und für problematieschere, wie solche mit den Lauten „h" auf der linken oder „C" bzw. „x" auf der rechten Seite habe ich außerdem im Wörterbuch bzw. in meinem persönlichen Bairischwortschatz nach Trägern gesucht.

Zwei Beispiele:

Für Diphone mit dem *Frikativ* „s" auf der linken Seite wurde der Träger „hamAs" (haben wir es) ausgewählt. Als Trägersatz wurde „danhObesgfrOgdhamAs"[3]... (Dann habe ich sie gefragt haben wir es...) verwendet. Für die Kombination „s-S"[4] wurde „SAfe" (Schemel) in den Trägersatz eingesetzt:

danhObesgfrOgdhamAsSAfe

Für den Diphon s-O das Wort „OvedsUA" (nach unten):

danhObesgfrOgdhamAsOvedsUA

usw.

Wenn das h der linke Teil eines Diphons ist, lässt sich, versucht man vorige Methode anzuwenden, kein Träger finden, der mit einem gesprochenen h endet. In diesem Fall sucht man sich für jeden möglichen Fall, sprich für alle Vokale und Diphthonge, ein Wort aus dem Wortschatz der Zielsprache:

h-e vONhebA (Wagenheber)

h-aI ghaIRAd(geheiratet)

[3]Der gesamte Satz muss nicht unbedingt Sinn ergeben.
[4]welche nur zwischen zwei Wörtern vorkommen kann

h-0A dAh0Am (daheim)

Bei dieser Methode muss man aber manchmal Kompromisse eingehen, in der Eignung der Wörter als Träger. So werden bei den Beispielen „geheiratet“ und „daheim“ die jeweiligen *Diphthonge* betont gesprochen. Durch einen bewusst monoton gesprochenen Trägersatz kann dieser kleine Mangel aber sehr gut wieder ausgeglichen werden.

5.2. Aufnahme

5.2.1. Ort

Die eigentliche Aufnahme sollte in einem Raum stattfinden, der so wenig wie möglich hallt, idealerweise in einer für solche Zwecke ausgelegten Sprecherkabine. Eine solche stand für die Aufnahmen zum bairischen Inventar leider nicht zur Verfügung. Die Aufnahmen wurden deshalb in einem kleinen Raum einer ruhigen Mietswohnung eines ruhig gelegenen Hauses gemacht. Es ist wichtig, dass keine vorbeifahrenden Fahrzeuge, radiohörenden Nachbarn und dergleichen die Qualität der Aufnahme mindern.

5.2.2. Material

Als Aufnahmegerät sollte möglichst ein DAT-Rekorder oder eine professionelle Soundkarte mit vorgeschaltetem Mikrofonvorverstärker verwendet werden. Außerdem mindestens ein qualitativ hochwertiges Mikrofon. Je mehr Mikrofone verwendet werden, desto mehr Auswahl an verschieden klingendem Quellmaterial hat man im Anschluss zur Verfügung, ohne die Aufnahme mehrmals wiederholen zu müssen. Man kann die Mikrofone unterschiedlich positionieren und bekommt dadurch teilweise sehr unterschiedliche Klangbilder der fertigen Aufnahme.

Die für diese Diplomarbeit erforderliche Aufnahme wurde mit einem *Shure sm58* Mikrofon durchgeführt, welches über einen externen Mikrofonvorverstärker an eine Soundkarte (*Terratec EWX 24/96*, 96 kHz, 24 bit) angeschlossen wurde.

Softwareseitig wurde der Open-Source-Waveeditor *Audacity* [5] verwendet.

[5] http://audacity.sourceforge.net

5.2.3. Testlauf

Vor der Aufnahme des gesamten Inventars ist es sinnvoll, einen einzelnen Satz zu transkribieren und nur alle darin vorkommenden Diphone aufzunehmen. Die fertig segmentierten und aufbereiteten (siehe Kap. 5.3, Seite 45) Sounddateien werden dann an die *MBROLA*-Leute gesendet, welche ein Testinventar erstellen, mit dem man nur diesen einzelnen Satz synthetisieren kann. Da das *MBROLA*-Team viel Erfahrung mit der Aufnahme und Verarbeitung der Datenbanken hat, bekommt man mit dem Testinventar auch Tipps zurück, wie die Qualität verbessert werden kann oder welche Probleme vielleicht auftreten könnten. Durch diese erste Rückmeldung wird unter Umständen eine Menge Arbeit gespart, welche ansonsten umsonst gemacht werden würde.

Der Testsatz, der von mir erstellt wurde, besteht aus 26 Diphonen und lautet folgendermaßen:

> `_i:sUAxUnsAmOIAnrOUdnTAgsIfarA_`
> (Ich suche uns einmal einen roten Taxifahrer.)

Der Satz wurde ausgewählt, um neben der allgemeinen Qualität auch einige, möglicherweise problematische, Diphone im Vorfeld begutachten zu können und, wenn nötig, andere Trägerworte zu verwenden. Im vorliegenden Satz sind deshalb die Diphthonge „`UA`“, „`OI`“ und „`OU`“ vertreten, außerdem das helle „`A`“ und das satte „`a`“. Alle diese Laute sind typisch für die bairische Sprache und gleichzeitig, wenn man von der Ähnlichkeit des bairischen satten `a´`s mit dem hochsprachlichen Normal-a absieht, im Deutschen und damit in den bisher existierenden Inventaren nicht vorhanden. Deswegen war der Satz ein guter Test, wieweit die fertige Datenbank tatsächlich „bairisch“ klingen würde.

Dazu war mir vor der Aufnahme nicht klar, ob ich die konsonantische Lautverbindung „`gs`“ als einen Laut oder als einen Diphon behandeln sollte. Würde sie als ein einzelner Laut berücksichtigt, wie das bei `ds`, `Z` und `pf` geschehen ist, müssten alle mit der Lautkombination möglichen Diphone mit ins Inventar integriert werden, was ein nicht zu verachtender Aufwand ist. Behandelt man sie jedoch als Diphon, bräuchte zwar nichts zusätzlich aufgenommen zu werden, allerdings könnte die Natürlichkeit der synthetisierten Sprache leiden. Wie sich nach dem Testlauf herausgestellt hat, war das aber nicht der Fall. Dazu kommt, dass das `gs` statistisch gesehen nicht so oft vorkommt, als dass man allzu große Einbußen der Gesamtqualität befürchten müsste.

5.2.4. Durchführung

Es ist zu empfehlen, vor der eigentlichen Aufnahme einige Tests mit verschiedenen Mikrofonpositionen durchzuführen, um einen möglichst idealen Klang der aufgenommenen Stimme zu bekommen. Nach erfolgreicher Positionierung folgt dann die eigentliche Aufnahme.

Die Wavedateien wurden einkanalig (Mono) mit der der Soundkarte höchstmöglichen Samplefrequenz von 96 kHz und einem Sampleformat von 16 bit/Sample aufgenommen. Die hohe Samplefrequenz ist zwar für das Endresultat nicht nötig, da das fertige Inventar mit einer Frequenz von 16 kHz ausgeliefert wird. Allerdings ist es nicht unwahrscheinlich, dass in naher Zukunft das Format der *MBROLA*-Inventare auf 32 kHz erweitert wird. Durch die Aufnahme mit 96 kHz wird zum Preis von mehr Plattenplatz, den die Aufnahme beansprucht, ein nochmaliges Aufnehmen und Segmentieren und somit eine Menge Arbeit eingespart. Die vorhandenen Schnittmarken lassen sich ohne weiteres auch mit auf 32 kHz downgesampelten Wavedateien wiederbenutzen.

Da die Aufnahme einige Zeit in Anspruch nimmt, es sind immerhin ca. 2000 Diphone und damit 2000 Trägersätze zu sprechen, sollte man darauf achten, dass der Sprecher genügend Pausen machen kann und immer etwas zu trinken bereitsteht, um seine Stimme nicht überzubeanspruchen. Es sollte allerdings versucht werden, die Aufnahme an einem einzigen Termin fertig zu stellen, da dadurch sichergestellt ist, dass sich nichts an der Aufnahmesituation geändert hat und die Konsistenz der gesamten Aufnahme gewährleistet ist. Notfalls ist es aber natürlich möglich, qualitativ schlechte oder schlecht gesprochene Aufnahmen in einer zweiten Sitzung zu wiederholen. Allerdings sollte man diese zusätzlich gemachten Aufnahmen markieren, damit dieser Umstand beim Erstellen des Inventars berücksichtigt werden kann.

In [tm00] wird außerdem empfohlen, vor jeder einzelnen Aufnahme den gleichen Ton abzuspielen. Ansonsten besteht die Gefahr, dass sich die Grundfrequenz der Stimme des Sprechers langsam ändert. Um dieser Gefahr vorzubeugen, wurde dem Sprecher, bei der Aufnahme zum Inventar ein synthetischer Ton per Kopfhörer eingespielt, an dem er sich orientieren konnte. Damit die Stimme möglichst natürlich klang, wurde im Vorfeld die durchschnittliche *f0*-Grundfrequenz des Sprechers ermittelt und dann zum Einspielen gewählt (in diesem Fall: 133 Hz).

Im Hinblick auf die Nachbearbeitung der Sounddateien ist es sinnvoll, diese geordnet zu erstellen. Es bietet sich an, alle Diphone mit dem gleichen Laut auf der linken Seite nach-

einander zu sprechen und in eine gemeinsame Datei aufzunehmen. Der Ordnung halber sollte außerdem immer die gleiche Abfolge der Laute auf der rechten Seite des Diphons benützt werden. Dadurch lässt sich die Segmentierung zumindest teilweise automatisieren (siehe Anhang D).

Da nicht jedes Trägerwort sofort perfekt gesprochen wird und deswegen unter Umständen wiederholt werden muss, ist es hilfreich, sofort jede befriedigende Aussprache eines Trägerwortes/-satzes akustisch zu markieren, um sie später ohne größere Umstände wiederzufinden. Praktisch wurde das im vorliegenden Fall durch ein kurzes Klopfen auf das Mikrofon nach jedem erfolgreich gesprochenen Trägersatz realisiert. Diese akustischen Zeichen lassen sich in den optischen Repräsentationen der Klangdateien, welche von Soundeditoren produziert werden, hervorragend wiederfinden.

5.3. Segmentieren

Vorbereitend müssen als Erstes alle Trägerworte in eigene Dateien extrahiert werden. Für diese erste mühselige Arbeit kann jeder beliebige Audioeditor verwendet werden.

Für die zweite mühselige Arbeit, das Segmentieren der Diphone, bieten sich zwei Programme an. Es gibt zum einen das *Diphone Studio*[6] (Abb. 5.1). welches ausschließlich dafür

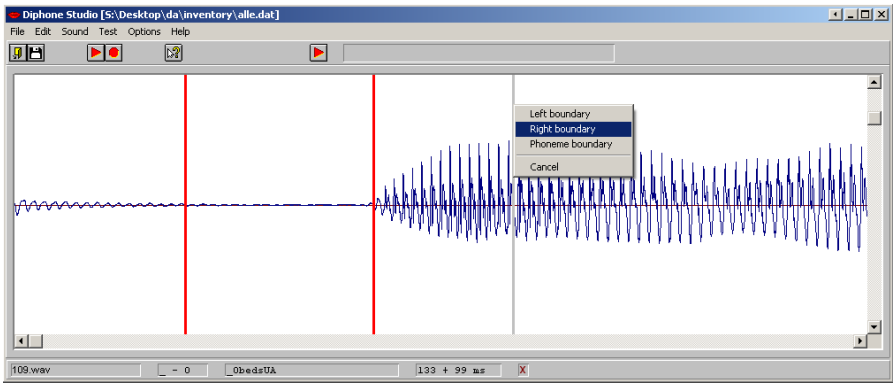


Abbildung 5.1.: Oberfläche von *Diphone Studio*

geschrieben wurde, das Segmentieren von Sounddateien für *MBROLA*-Inventare zu erleichtern und zu beschleunigen. Das Programm erwartet als Eingabe eine Textdatei mit Zeilen der Form:

[6] im Internet unter `http://www.fluency.nl/dstudio/dstudio.htm` zu finden

```
..
_    I    w7.wav    _IrmengAd
_    o:   w8.wav   _o:bsdSoIn
_    O    w9.wav    _ObezUA
..
```

Die erste Spalte kennzeichnet den linken Teil des Diphons (hier die Stille), die zweite den rechten Teil. Die dritte Spalte ist der Dateiname des Soundfiles, das das Trägerwort enthält und die vierte (nicht unbedingt notwendige) Spalte zeigt die Transkription des Trägerwortes.

Beim Segmentieren muss jeder Diphon mit drei Markierungen versehen werden. Die erste Markierung wird in der Mitte des linken Teillautes gesetzt, die Zweite analog dazu in der Mitte des rechten Teillautes. Die dritte Markierung wird dazu benutzt, beide Laute voneinander zu trennen und wird dementsprechend am Lautübergang gesetzt.

Diphone Studio bietet über das Kontextmenü eine Auswahl der zu setzenden Markierungen und ermöglicht so eine zügige Bearbeitung des gesamten Datenbestandes. Ein Nachteil ist allerdings, dass die Software außer der Wellenform des Signals keine weiteren optischen Repräsentationen des Klanges liefert. Dadurch lassen sich Lautgrenzen, vor allem zwischen Vokalen, manchmal nur sehr schwer und ungenau finden.

Mit der Verwendung von *Praat*(Abb. 5.2, Seite 47) als Segmentierhilfe kann man dieses Problem umgehen. In Praat lässt sich neben der Wellenform des Signals wahlweise auch dessen *Spektrogramm*, der *f0*-Wert und die Formanten anzeigen. Alle diese Darstellungen sind bei der Segmentierung wertvoll. Vor allem der Formantverlauf zeigt Lautwechsel sehr deutlich an. Leider ist Praat von sich aus nicht für das schnelle, massenweise Segmentieren von Diphonen geeignet. Vor allem das Eingeben der Bezeichnungen von Diphonen und das damit verbundene Wechseln zwischen Maus und Tastatur benötigt Zeit. So dauert das Markieren eines Diphons schätzungsweise doppelt solange wie die entsprechende Aktion im *Diphone Studio*. Allerdings lässt sich dieser Mangel teilweise durch das Schreiben eines Skriptes beheben. Die Möglichkeit der Einbindung von Skripten macht aus dem sowieso schon sehr mächtigen Werkzeug *Praat* ein noch Mächtigeres. Das Skript, welches ich zur Unterstützung der Segmentierung von Diphonen geschrieben habe und die daraus resultierende Vorgehensweise beim Segmentieren beschreibe ich in Anhang D.

Bevor man die fertig segmentierten Daten dann zur abschließenden Erzeugung des Inventars abschickt, müssen sie noch in Form gebracht werden. Das vom *MBROLA*-Team verlangte Format des Datenbestandes sieht folgendermaßen aus:

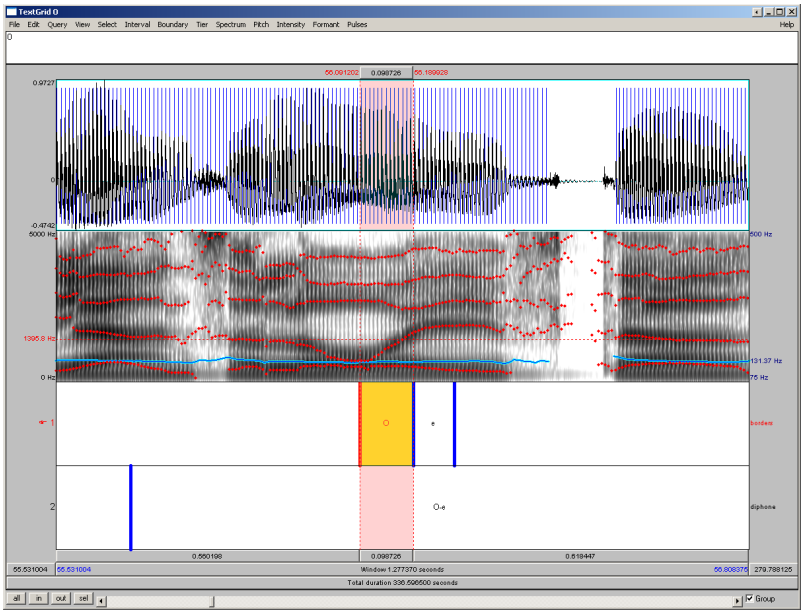


Abbildung 5.2.: Praat

- Pro Diphon eine Sounddatei, wobei sich vor der ersten bzw. nach der dritten Markierung noch mindestens 600 Samples befinden sollten, da der Kontext des Diphons bei der Berechnung des Inventars mit einbezogen wird, und...
- eine Textdatei (mit der Endung `.dat`), in welche für jeden Diphon der Name der entsprechenden Sounddatei, der jeweilige linke und rechte Laut sowie die Positionen[7] der drei Markierungen (in der Reihenfolge: linke Diphongrenze, rechte Diphongrenze, Mitte) stehen muss.

Eine korrekte Zeile einer solchen Datei sieht demnach z. B. so aus:

```
w8.wav    _    0    1000    4705    3128
```

In der Praxis wurde der Großteil des Segmentierens mit *Praat* erledigt. Zur Nachbearbeitung in Einzelfällen und zum Testen und anschließendem Exportieren des gesamten Datenbestands in das vorgeschriebene Format wurde dann *Diphone Studio* verwendet, da es hierfür einige nützliche Funktionen bereitstellt. Zum Testen kann man sich z. B. eine Folge von Lauten vorlesen lassen. Die entsprechenden Diphone werden dazu zwar nur unbearbeitet nacheinander abgespielt, aber um einen ersten Eindruck zu gewinnen und Diphongrenzen grob zu testen ist diese Funktion sehr nützlich. Beim Exportieren in das

[7] in Samples gezählt

vorgeschriebene Datenformat hilft *Diphone Studio*, indem es die geforderte `.dat`-Textdatei automatisch erstellt und außerdem die Sounddateien auf jeweils 1000 Samples vor der ersten und ebenso viele Samples nach der letzten Markierung stutzt. Diese Maßnahme verkleinert den Umfang des gesamten hochzuladenden Pakets beträchtlich[8].

[8] bei mir von ca. 75 mb auf 16 mb

6. Erstellung der Regeln für die Umsetzung

Alle folgenden Regeldateien und Wörterbucher werden nach ihrer Fertigstellung mithilfe der Software *Sigmalab*[1] kompiliert bzw. zusammengefaßt. *Sigmalab* stellt für diesen Zweck eine Skriptsprache bereit.

6.1. Der Regelsatz

Da es anscheinend keinen Regelsatz gibt, der versucht, geschriebene Hochsprache in bairische Phoneme umzusetzen, habe ich versucht, ausgehend vom bestehenden Hochdeutschen, einen solchen zu erstellen.

6.1.1. Wörter

Die Regeln für die Umsetzung von Buchstaben in Phoneme werden in eine einfache Textdatei geschrieben, welche anschließend mittels *Sigmalab* kompiliert wird. Die Einträge sind in der folgenden Form enthalten:

```
#       a       #       A
..
.       alt     .       OId
..
K       en      #       n
.       en      .       A
..
#       ge      K       ge
#       ge      V       g?"
#       ge      .       g
```

[1] `http://home.t-online.de/home/gesim/si_p1_d.htm`

```
..
ge      h       .       h
..
```

In der ersten Spalte wird der linke Kontext des zu bearbeitenden Kerns angegeben. In der Zweiten der Kern selbst und in der dritten Spalte dann der rechte Kontext. Die vierte Spalte gibt an, als welche *SAMPA*-Sequenz der Kern, im Falle einer Übereinstimmung aller drei Voraussetzungen, interpretiert werden soll.

Der linke und rechte Kontext kann aus den Zeichen des verwendeten Alphabets bestehen, außerdem den beiden Sonderzeichen #[2] und .[3] sowie selbst definierten Gruppen von äquivalenten Zeichenfolgen. Die Zusammensetzung der Letzteren müssen beim Kompilieren durch *Sigmalab* angegeben werden. Ein Beispiel für eine solche Definition ist:

```
D     {äu|au|ai|ay|ei|ey|eu}
```

Hier wird das *D* als Platzhalter für eine beliebige Vokalkombination aus äu,au,ai,ay,ei,ey und eu festgelegt. Erforderlich ist lediglich, dass der Platzhalter nicht schon im Alphabet benützt wird.

Beschränken sich die äquivalenten Zeichengruppen auf jeweils nur ein Zeichen, erfolgt die Definition ein klein wenig anders, nämlich anhand sog. äquivalenter Paare:

```
Va Ve Vi Vo Vu Vy Vä Vö Vü
Kb Kd Kg Kp Kt Kk
```

Das erste Zeichen wird hier innerhalb der Regeln durch das Zweite ersetzt.

Die Regeln werden Zeile für Zeile von oben nach unten abgearbeitet. Passt eine Regel auf den Anfang des gesuchten Wort(teil)es, dann wird der Kern durch den Inhalt der vierten Spalte ersetzt und für den Rest des Wortes werden wieder alle Regeln bis zu einem Treffer durchlaufen.

In dem Beispiel oben würde das Wort *gehalten* folgendermaßen ersetzt werden:

```
Durchlauf                       Rest      Resultat
-------------------------------------------------
1           ge    ->    g       halt      g
2           halt  ->    hOId    en        ghOId
3           en    ->    n                 ghOIdn
```

[2] welches die vordere bzw. hintere Wortgrenze markiert
[3] wofür jeder beliebige Buchstabe eingesetzt werden kann

Zum Erstellen des bairischen Regelwerkes habe ich ein Perl-Skript (Anhang C) geschrieben, welches den oben beschriebenen Ablauf nachbildet. Damit konnte jede Regel sofort und einfach getestet werden. Wird ein Wort falsch umgesetzt, ermöglicht das Skript, per Kommandozeilenoption „`-d`“ eine Debug-Ausgabe. Diese enthält die Zeilennummer, in welcher die Regel steht, welche zur Übersetzung verwendet wurde. Eine beispielhafte Ausgabe (eines richtig Übersetzten Wortes) sieht so aus:

```
$ trans.pl -r bay.txt -d gold
line nr.: 792    word:
match: g
sub: g
new rest: old
line nr.: 1131   word: g
line nr.: 1146   word: g
line nr.: 1147   word: g
match: ol
sub: oI
new rest: d
line nr.: 351    word: goI
line nr.: 362    word: goI
line nr.: 369    word: goI
match: d
sub: d
new rest:
gold    -->     goId
```

Die Kommandozeilenoption „`-r`“ ist zwingend erforderlich und gibt die Datei, welche die Regeln enthält, an. Jede Regel, bei der der Kern und der linke Kontext mit dem Wort übereinstimmen, wird in der Form
„`line nr.:` *zeilennummerDerRegel* `word:` *bisherigGebildeteUebersetzung*
ausgegeben. Stimmen der Kern und beide Kontexte überein, wird in der nächsten Zeile der Kern ausgegeben. In der nächsten Zeile folgt die Ersetzung und danach der neue, noch zu bearbeitende Reststring. Das Beispiel hier zeigt die Übersetzung von „Gold“ in „`goId`“(mit dem Diphthong „oI“!).

Die praktische Vorgehensweise war, dass ich als Erstes, ausgehend von der hochdeutschen Umsetzung, geprüft habe, ob sich die Vorgaben der Regel auch auf bairische Gegebenheiten umsetzen lassen. Dafür habe ich ein kurzes Shell-Skript geschrieben, welches mithilfe von *Regular Expressions* alle Wörter entweder aus dem kleinen oder dem großen hochdeutschen Lexikon heraussucht, für das die Vorgaben[4] passen. Damit konnte ich leichter erkennen, ob sich der Kern in allen diesen Wörtern gleich umsetzen lässt. War das der Fall, wurde die Regel geändert. Ansonsten wurde versucht, die Vorgaben der Regel so zu verfeinern, dass sich sinnvolle Regeln aufstellen ließen.

6.1.2. Zahlen

Das Erstellen der Regeln für die Umsetzung von Zahlen und Daten ist eine aufwendige und nicht unkomplizierte Arbeit. Im vorliegenden Fall konnte allerdings auf die bestehende, hochdeutsche Grammatik zurückgegriffen werden, da sie sich fast nicht von der bairischen unterscheidet. So sind lediglich die elementaren Teile, aus denen zusammenhängend ausgesprochene Zahlen (etwa „zwei“ „und“ „dreißig“) in die entsprechende bairische Form zu übersetzen. Die Art und Weise, wie die Teile zusammengesetzt werden, ändert sich nicht. Im Anschluß ein kleiner Ausschnitt aus der entsprechenden Datei „`num_bai.txt`“:

```
<1.n>   1       eASdn
<1.s>   1       eASds
<1.ns>  1       eASdns
<2>     2       dsvOA
```

6.2. Das Wörterbuch

Das Übersetzten des Wörterbuches verursacht auch keine größeren Schwierigkeiten, sieht man davon ab, dass allein für das kleine Lexikon mit den statistisch am häufigsten verwendeten Begriffen ca. 5100 Wörtern die entsprechende bairische Aussprache zugeordnet werden muss. Das Format des als einfacher Textdatei abgespeicherten kleinen Lexikons sieht folgendermaßen aus:

```
allzu   "OIdsU    AV
```

[4] d. h. linker Kontext - Kern - rechter Kontext

```
als     "OIs      PR,CO
also    "OIso:    PA
alt     "OId      AJ
```

Es wird, wie bereits im Kap. 2.2.2 kurz erwähnt, neben der lautlichen Entsprechung[5] auch die Wortart des Eintrages angegeben. Im Gegensatz zum nachgeschalteten großen Lexikon, wo diese Angabe fehlt.

Um ein Bairisch als Ausgabe zu produzieren, das dem nahe kommt, welches heute gesprochen wird, wurde beim Erstellen des Wörterbuchs bewußt darauf verzichtet, urbairische Wörter zu verwenden, die heute nicht mehr verwendet, manchmal auch nicht mehr verstanden werden. Ein Beispiel sind die Wochentage, welche früher z. B. „`IRdA`“ (Dienstag), „`pfInsdA`“ (Donnerstag) oder „`sUndA`“ (leichter nachzuvollziehen: Sonntag) genannt wurden. Aber selbst ältere Leute benutzen heute eine abgeschwächte bairische Version der Hochdeutschen Bezeichnungen (z. B. „`do:nAsdOg`“ für „Donnerstag“).

Allerdings gibt es sehr wohl hochsprachliche Wörter, für die sich im Bairischen keine ähnlich klingenden Entsprechungen gibt. Diese wurden selbstverständlich mit dem bairischen Pendant übersetzt. Einige Beispiele: „`SmaRn`“ (Quatsch), „`bUAm`“ (Knaben), „`fi:C`“ (Tier)[6].

[5] mit Betonung
[6] allerdings „`tIAbARg`“ (Tierpark) und „`tIAhaIm`“ (Tierheim)

7. Probleme / Einschränkungen

7.1. Grammatik / Wortschatz

Eine Schwierigkeit bei dem Vorhaben, Übersetzungen für hochdeutsche Wörter zu finden, ist der Umstand, dass oft mehrere verschiedene davon möglich sind. Von denen auch alle im täglichen Sprachgebrauch verwendet werden. Und nicht nur durch zwei verschiedene Sprecher. Oft verwendet derselbe Sprecher die unterschiedlichen Varianten, ohne dass man eine Regel erkennen könnte, warum das geschieht. Vor allem durch Wörter, die durch den hochsprachlichen Einfluss dem Bairischen hinzugefügt wurden, entstehen mehrere Möglichkeiten der Aussprache. Ein Beispiel ist das Wort „Treppe“. Man verwendet entweder die ursprüngliche, bairische Version „SdIAN“, oder die neuere, „drEbm“. Es kommt auch vor, dass die eher hochdeutsche Version dann verwendet wird, wenn etwas nicht ganz ernst zu nehmen ist. „Du bist eine ganz Hübsche.“[1] kann mit „dU bisd A gans A **hIbSe.**“ übersetzt werden. Dann ist die Betreffende vielleicht eine ein wenig zu eitle Frau oder ein kleines Mädchen. Sagt man hingegen zu ihr „dU bisd A gans A **saUbAne.**“, drückt man ehrlichen Respekt ob der Schönheit der Angesprochenen aus.

Ein anderes Beispiel für eine Situation, wo nicht so ohne Weiteres entscheiden werden kann, wie die Übersetzung lauten soll, ist der Satz „Sollen wir laufen?“:

- „soIn m*IA* lAfA“
  fragt, ob gerade *wir* laufen sollen, wohingegen
- „soIn m*A* lAfA“
  die Frage danach ist, ob gelaufen werden soll, oder vielleicht doch lieber gegangen.

Die Betonung im Satz, auf „wir“ im ersten und „laufen“ im zweiten Fall, unterscheidet sich hingegen nicht von der hochdeutschen Aussprache.

Alle wesentlichen grammatikalischen Unterschiede (Kap. 3.3) sowie den gesamten bairischen Wortschatz nachzubilden, ist im Rahmen dieser Diplomarbeit leider nicht möglich.

[1] In Wirklichkeit sagt man natürlich wohl eher: „Du bist sehr hübsch.“

TFHTTS stellt zudem auch keine Möglichkeit bereit, selbst wenn es manche Wortarten richtig erkennt, syntaktische Gebilde umzuformen und so z. B. die 1. Vergangenheit in die 2. zu wandeln.

7.2. Prosodie

Es hätte die zeitlichen Vorgaben dieser Diplomarbeit gesprengt, auch die prosodischen Merkmale der bairischen Mundart zu untersuchen und nachzubilden. Ich habe leider, außer in [Kuf61], wo die Prosodie der Münchner Stadtmundart beschrieben wird, nur wenig zum Thema gefunden.

8. Schlusswort

Es hat sich herausgestellt, dass es nur sehr bedingt möglich ist, die bairische Aussprache anhand der hochdeutschen Schriftsprache vorherzusagen. Es gibt einfach zu viele Ausnahmen und Sonderfälle, die ja mit den Charme des bairischen Dialektes ausmachen. Ich denke, würde man alle Wörter, die im großen Wörterbuch enthalten sind, manuell in *SAMPA* umsetzen, würde sich die Qualität der Ausgabe um einiges verbessern.

Auch müsste die Dauersteuerung und die Prosodiegenerierung von *TFHTTS* besser an die Gegebenheiten der bairischen Sprache angepasst werden, als mir das im Rahmen der Diplomarbeit möglich war.

Die Ergebnisse dieser Diplomarbeit lassen sich deshalb in etwas so zusammenfassen:

TFHTTS ist nun in der Lage, die Synthesefähigkeit und die große Auswahl an Inventaren von *MBROLA* zu benutzen. Es wurde ein neues *MBROLA*-Inventar erstellt, welches durchaus befriedigende Ergebnisse liefert und welches auf der Hompage von *MBROLA* von jedem zur freien Benutzung heruntergeladen werden kann. Beim Versuch, die bairische Mundart für das Synthesesystem *TFHTTS* aufzubereiten, gab es einige Probleme, die zu lösen es mehr Zeit gekostet hätte, als zur Verfügung stand. Nichtsdestotrotz bietet *TFHTTS* nun die Option der bairischen Ausgabe. Diese ist, gibt man den zu bearbeitenden Text in einer der bairischen Grammatik entsprechenden Form ein, auch durchaus als bairisch zu erkennen.

Es ergeben sich aus den Resultaten der Diplomarbeit eine menge Ansatzpunkte, die aufzugreifen und weiterzuverfolgen sehr interessant wäre. Vor allem die noch nicht bearbeitete Frage der Prosodie näher zu untersuchen und das Umsetzen vieler grammatikalischen Eigenheiten scheinen sehr spannend und lohnend.

A. SAMPA-Liste der deutschen Laute

SAMPA	Beispiel	SAMPA	Beispiel
_	Stille, kein Laut	R	Riese, Kraut
p	Pier, ab	6	Oper, der
b	Bier, Krabbe	j	jetzt, Jagd
t	Tier, Grad	aI	eins, Kaiser
d	dir, edel	OY	Äußerung, neu
k	Kasse, Tag	aU	auf, schau
g	Gasse, egal	@	sehen, besagt
f	Vogel, Schlaf	i:	Igel, bieten
v	Wasser, eventuell	I	in, bitten
s	Ast, Hass	y:	Übung, hüten
z	Sieb, Besen	Y	Ypsilon, Hütten
S	spät, Asche	e:	beten, Schnee
Z	Genie, Dschungel	E	Betten, Gäste
x	nach, doch	E:	Äsen, Gebläse
C	dich, Honig	2	Öfen, mögen
h	Hut, Ahorn	9	öffnen, können
pf	Pferd, Topf	u:	buhlen, gut
ts	zwei, Platz	U	lustig, Butter
tS	Kutsche, Cello	o:	Ofen, Kohl
m	Mut, Hammer	O	offen, Topf
n	Nase, Kanne	a:	war, wahr
N	eng, bange	a	an, kann
l	Liebe, Halle		

B. Die Klasse `MbrolaTranslation`

```
// MbrolaTranslation.cpp: Implementierung der Klasse CMbrolaTranslation.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "MbrolaTranslation.h"
#include "StdAfx.h"
#include "globals.h"
#include "into_mod.h"
#include "Synthese.h"

//////////////////////////////////////////////////////////////////////
// Konstruktion/Destruktion
//////////////////////////////////////////////////////////////////////

CMbrolaTranslation::CMbrolaTranslation()
{}

CMbrolaTranslation::~CMbrolaTranslation()
{}

short CMbrolaTranslation::dat_aus_pho(struct fuji_data *p_fdat, char *in,
char *out, short phonem_count)
{
 char buffer[1024]; //zeile
   char *pfstr;    // grundfrequenz
   short line_count, ds_zaehler, l;
   int dec, sign, skipNextPhonem;
```

```
struct fuji_data *p_fdat1;
unsigned int anfang=0;
float time;
char time_str[10];

float f0_pitch = GetGFFactor();

line_count = 0;
ds_zaehler = 0;
skipNextPhonem = 0;
while( (anfang=lese(buffer,in,anfang)) <= strlen(in) )
{
 if ( buffer[0] == '{' && check_clause_intona (buffer)!=-1)
  {
   continue;
       }
 else
  {
   if ( line_count++ == 4 )
    {
     line_count = 1;     /* internen Zaehler ruecksetzen */
              ds_zaehler++;     /* Datensatzzaehler erhoehen */
              continue;
            }
   // Ab hier wird fuer mbrola angepasst
         if ( line_count == 2 )
    {
     if (skipNextPhonem != 2)
      {
       char *token;
                  token = strtok ( buffer, "," );
                  //phonem
                  if ( token[0]=='_' && token[1]!='\0' )
       {
        token = &token[1];
```

```
                    skipNextPhonem = 2;
                  }
    strcat (out, token );
                strcat (out, " " );
                token = strtok (NULL, ",\n" );
                strcat (out, token );
                // dauer
                strcat (out, " " );
              }
  else
   {
    skipNextPhonem = 1;
             }
 }
if ( line_count == 3 ) /* Grundfrequenzzeile wird generiert */
        {
  if (skipNextPhonem != 1)
   {
    buffer[0]='\0';
                p_fdat1 = p_fdat + ds_zaehler;
                for (l = 0; l < GF_PNT; l++)
     {
      time = (float)l*(1.0f/(float)GF_PNT);
                    sprintf(time_str,"%3.0f",time*100);
                    strcat (buffer, time_str );
                    strcat(buffer," ");
                    pfstr = ecvt ((double)p_fdat1->gf[l] *
                    TTS_DEFAULT_PITCH*f0_pitch, 4, &dec, &sign);
                    strncat (buffer, pfstr, dec );
                    strcat (buffer, " " );
                  }
    strcat (buffer, " \n" );
                if(p_fdat1->number == phonem_count)
     {
      buffer[strlen(buffer)-2]=0;
```

```
                        strcat (buffer, "100" );
                        strcat(buffer," ");
                        pfstr = ecvt ((double)p_fdat1->gf[GF_PNT-1] *
                        TTS_DEFAULT_PITCH*f0_pitch, 4, &dec, &sign);
                        strncat (buffer, pfstr, dec );
                        strcat (buffer, "\n" );
                      }
       }
      else
       {
        skipNextPhonem = 0;
                 }
      if(strncmp (buffer, "_,", 2 )== 0 )
       strcpy (buffer, "\n\0"); /* kein GF-Wert fuer Pausen */
              strcat(out, buffer);
            }
   }

  }
 return(1);
}
```

C. Das Perl-Skript `translate.pl`

```
#! /usr/bin/perl

# programm, um woerter eines lexikons ins bayrische zu uebersetzen
#
# 10. Mai 2004, Markus Binsteiner
#
# usage: trans.pl filename ruleset [ruleNr]

use File::Basename;
use Getopt::Std;

$start = time;
$prog=basename $0;
$syntax = "Syntax: $prog [-d] [-n rulenumber]
     [-l lexicon] -r ruleset [word [word ...]]\n";

getopts('dl:n:r:') or die $syntax;

($opt_r) or die $syntax;

local $debug = $opt_d;

local @RULES;
local $ruleToExecute = $opt_n;

$rulesfilename = $opt_r;
(-f $rulesfilename && -r $rulesfilename ) or
 die "$prog: $opt_r: Datei nicht vorhanden oder nicht lesbar\n";
```

```
$filename = $opt_l;
if ( $filename ) {
(-f $filename && -r $filename ) or
 die "$prog: $filename: Datei nicht vorhanden oder nicht lesbar\n";
} else {
#if ($ARGV[0] !~ /^[a-z]+$/) {
#  die "$prog: $filename: word has to be all lowercase\n";
#} else {
  @words = @ARGV;
#}
}
# load rules
open(RULEFILE, "< $rulesfilename")
  or die "Could not open $rulesfilename for reading: $!\n";

while (<RULEFILE>) {
  if ( $_ !~ m/^\s/ ) {
    push(@RULES, $_);
  }
}
close(RULEFILE);

if ( $words[0] ) {
  foreach $oneword (@words) {
    translate ( $oneword );
}
} else {
  # open the lexicon
  open(LEXICON, "< $filename")
    or die "Could not open $filename for reading: $!\n";

  # go through all lines
  while (<LEXICON>) {
    translate ($_);
  }
```

```
  close(LEXICON);
}

$stop = time - $start;
print "stop: $stop\n";


# uebersetzen
sub translate {
  ($word, $sampa, $prop) = split /\t/, $_[0];
  $wordOld = $word;

  $leftOfWord = "#";
  $rest = "$word";
  #print "wort vorher: $left$word\n" if $debug;
  $word = "";

  LINE: while ( $rest ) {
  #print "new rest\n" if $debug;
  $nr = 0;
  # go through each rule
    foreach $rule (@RULES) {
    $nr = $nr+1;
    ($left, $match, $right, $sub) = split /\s+/, $rule;

      # execute rule at all?
      #if ( $match eq $rest ) {
      #print "debug: execute rule nr. $nr\n" if $debug;
      #$left =~ s/#/\^#/;
      $left =~ s/\./(^|.*)/;
      $right =~ s/#/\$/;
      $right =~ s/\./(.*|\$)/;

      $match =~ s/A/ä/g;
      $match =~ s/6/ö/g;
```

```
$match =~ s/Y/ü/g;
$match =~ s/2/ß/g;

# replace konstants
$left =~ s/D/(äu|au|ai|ay|ei|ey|eu)/g;
$left =~ s/C/(X|ch|ph|qu|sch)/g;
$left =~ s/T/(bb|ck|dd|ff|gg|kk|ll|mm|nn|pp|rr|ss|tt)/g;
$left =~ s/Q/(abel|al|alis|ant|anz|ärin|ator|
ell|ent|enz|et|eur|iant|ibel|iell|ient)/g;
$left =~ s/Z/(in|ion|ismus|ist|istik|istin|itis|iv|ivum)/g;
$left =~ s/S/(Q|Z)/g;
$left =~ s/N/(chen|ler|lein|lich)/g;
$left =~ s/O/(ien|ik|isch|ium|ius|um|ung)/g;
$left =~ s/U/(ier)/g;
$left =~ s/E/(e|em|en|es|er|ern|n|nen|s|ere|erem|
eren|erer|eres|ste|sten)/g;
$left =~ s/I/(e|en|est|et|ete|eten|etest|etet|n|t|
te|ten|test|tet)/g;

$left =~ s/A/ä/g;
$left =~ s/6/ö/g;
$left =~ s/Y/ü/g;
$left =~ s/2/ß/g;

$left =~ s/V/\[aeiouyäöü\]/g;
$left =~ s/B/\[aou\]/g;
$left =~ s/X/\[bcdfghjklmnprstvwxz\]/g;
$left =~ s/P/\[bdg\]/g;
$left =~ s/K/\[bdgptk\]/g;
$left =~ s/L/\[lmnr\]/g;

$right =~ s/D/(äu|au|ai|ay|ei|ey|eu)/g;
$right =~ s/C/(X|ch|ph|qu|sch)/g;
$right =~ s/T/(bb|ck|dd|ff|gg|kk|ll|mm|nn|pp|rr|ss|tt)/g;
$right =~ s/Q/(abel|al|alis|ant|anz|ärin|ator|ell|
```

```
        ent|enz|et|eur|iant|ibel|iell|ient)/g;
      $right =~ s/Z/(in|ion|ismus|ist|istik|istin|itis|iv|ivum)/g;
      $right =~ s/S/(Q|Z)/g;
      $right =~ s/N/(chen|ler|lein|lich)/g;
      $right =~ s/O/(ien|ik|isch|ium|ius|um|ung)/g;
      $right =~ s/U/(ier)/g;
      $right =~ s/E/(e|em|en|es|er|ern|n|nen|s|ere|erem|
       eren|erer|eres|ste|sten)/g;
      $right =~ s/I/(e|en|est|et|ete|eten|etest|etet|n|t|
       te|ten|test|tet)/g;

      $right =~ s/A/ä/g;
      $right =~ s/6/ö/g;
      $right =~ s/Y/ü/g;
      $right =~ s/2/ß/g;

      $right =~ s/V/\[aeiouyäöü\]/g;
      $right =~ s/B/\[aou\]/g;
      $right =~ s/X/\[bcdfghjklmnprstvwxz\]/g;
      $right =~ s/P/\[bdg\]/g;
      $right =~ s/K/\[bdgptk\]/g;
      $right =~ s/L/\[lmnr\]/g;


      # match a rule?
      if ($rest =~ /^$match($right.*)/ ) { # treffer ?
       $newRest = $1;
       print "line nr.: $nr\t  word: $word\n" if $debug;
      if ( $leftOfWord =~ /$left$/ ) {
$rest = $newRest;
print "match: $match\n" if $debug;
print "sub: $sub\n" if $debug;
$word = "$word$sub";
$leftOfWord = "$leftOfWord$match";
print "new rest: $rest\n" if $debug;
```

```
next LINE;
}
      }
    }

  $word = "$word--$rest";
  $rest = "";
  next LINE;

}
  chop ($prop);
  print "$wordOld\t->\t$word\t$prop\n";
}
```

D. Praat-Skript als Segmentierhilfe

Das folgende Skript hilft beim Segmentieren von Diphonen, die in einer Wave-Datei abgelegt sind und eine bestimmte Ordnung aufweisen. Diese Ordnung ist innerhalb des Skripts vorgegeben.

Zur Anwendung: Man markiert in Praat das Trägerwort des Diphons und führt das Skript aus. Dann wird automatisch in die Auswahl gezoomt und die Markierungen samt Beschriftung der Diphonteile eingefügt. Alles was noch zu tun bleibt, ist die drei Markierungen zum Markieren des Diphons an die richtige Stelle zu verschieben.

In seiner jetzigen Form ist das Skript nur für meine Bedürfnisse bei der Segmentierung der bairischen Sprachdaten zugeschnitten. Es lässt sich jedoch sehr leicht abändern und anpassen.

```
begin = Get begin of selection
end = Get end of selection
length = Get selection length

middle = begin + ( length / 2 )

Move cursor to... begin

Add on tier 2
Move cursor to... end
Add on tier 2

endeditor
sound = selected ("Sound")
textgrid = selected ("TextGrid")

namePhone1$ = selected$ ("TextGrid", 1)
```

```
select textgrid
last = Get number of intervals... 2

last = last-3

labelBefore$ = Get label of interval... 2 last

!printline labelbefore ’labelBefore$’

label1$ = "_"
label2$ = "A"
..
hier sind in der Richtigen Reihenfolge alle Phone aufgeführt.
..
label44$ = "t"
label45$ = "v"
label46$=""
label47$="_"

i=1
while labelBefore$ <> label’i’$
i = i + 1
endwhile

i = i + 1

!printline i ’i’
lastLabel$ = label’i’$

!printline namePhone ’namePhone1$’
!printline lastlabel ’lastLabel$’

lastBegin =  Get starting point... 2 last+2
lastEnd = Get end point... 2 last+2
lastMiddle = lastBegin + (lastEnd - lastBegin)/2
```

```
Insert boundary... 1 lastMiddle-0.2
Insert boundary... 1 lastMiddle
Insert boundary... 1 lastMiddle+0.2

last2 = Get number of intervals... 1
Set interval text... 1 last2-2 ’namePhone1$’
Set interval text... 1 last2-1 ’lastLabel$’
Set interval text... 2 last+2 ’lastLabel$’

Write to text file... ’namePhone1$’.temp

plus sound
editor

Zoom... begin end
```

Glossar

Affrikat Ein Plosiv, der in einen Frikativ übergeht.

Allophon Eine von mehreren phonetischen Repräsentationen eines Phonems. Die Bedeutung einer Äußerung ändert sich durch einen Austausch von Allophonen nicht. Allophone im Deutschen sind z. B. das stimmhafte und das stimmlose ‘s’.

alveolar Die Ausprache eines alveolaren Lautes erfolgt am oberen Zahndamm.

Cygwin Eine Sammlung von Programmen, welche unter dem Betriebssystem Windows eine unixartige Umgebung bereitstellen.

dental Bei der Artikulation dentaler Laute sind die Zähne beteiligt.

Diphthong Eine Folge von zwei vokalischen Lauten hintereinander, wobei der erste Laut in den zweiten übergeht. Im Deutschen gibt es die drei Diphthonge ‘au’, ‘ei’, ‘eu’

Formanten Charakteristische Frequenzbereiche eines Signales. Entstehen bei der Sprache durch Resonanzbereiche im Vokaltrakt.

Frikative Reibelaut, entsteht durch Verengung des Artikulationskanals.

glottal Die Aussprache eines Lautes, wobei die Stimmritze der Glottis als Artikulationsort dient.

Graphem Die kleinste Einheit einer Schriftsprache.

Homograph Wörter, die gleich geschrieben werden, aber verschieden ausgesprochen und gedeutet werden, z. B. die Heroin und das Heroin.

labial Der Artikulationsort labialer Laute sind die Lippen.

Morphem Das kleinste bedeutungstragende Element einer Sprache.

palatar Die Aussprache palatarer Laute bedingen einen Kontakt zwischen Zunge und Palatum

Phonem Kleinste bedeutungsunterscheidende Einheit eines Sprachsystems.

Plosiv Konsonanten, bei deren Artikulation die Luft erst blockiert und danach 'explosionsartig' wieder freigegeben wird.

Prosodie Die Prosodie beschreibt die Betonung, den Rhythmus und die Melodie einer Sprache.

Regular Expressions Eine Gruppe von formalen Programmiersprachen, welche ein sehr leistungsfähiges und mächtiges System zur Stringbearbeitung und -manipulation bereitstellen.

SAMPA Speech Assessment Methods Phonetic Alphabet. Phonetisches Alphabet zur Verarbeitung von Sprachen durch Computer.

Sonorant Gruppenbezeichnung für Liquide/Laterale, Nasale

Spektrogramm Die Darstellung der Frequenzen eines Signals im zeitlichen Verlauf.

symbolisch linguistische Repräsentation Eine Repräsentation einer Äußerung, welche Informationen über die lautlichen Eigenschaften der Äußerung mittels Symbolen transportiert.

Unicode Internationaler Standard, um jedes graphische Zeichen aller bekannten Schriftkulturen und Zeichensysteme in einem Zeichensatz unterzubringen.

uvular Die Aussprache eines Lautes unter Beteiligung des Gaumenzäpfchens.

velar Bei der Aussprache velarer Laute kommt es zu einem Kontakt von Zunge und Velum

Literaturverzeichnis

[Gib98] Dafydd Gibbon. *Grundkurs Linguistik*. Universität Bielefeld, 1998. http://www.spectrum.uni-bielefeld.de/ Classes/Summer98/Grundkurs98/Vorlesung/ grundkursvorlesung/. 3

[GK04] Barbara Madella-Mella Günther Kreuzbauer. *Die paraverbale Komponente in der Praxis des Rhetoriktrainings*. Institut für Geschichte Universität Salzburg, Arbeitsgruppe Rhetorik, 2004. http://www.rheton.sbg.ac.at/articles/01.04/kreuzbauer.pdf, zuletzt gesichtet am 16. Juni 2004.

[Kla79] D. H. Klatt. Synthesis by rule of segmental durations in english sentences. *Frontiers of Speech Communication Research*, pages 287 – 299, 1979. New York. 10, 11

[Kuf61] Herbert L. Kufner. *Strukturelle Grammatik der Münchner Stadtmundart*. R. Oldenbourg, München, 1961. 26, 55

[Mer75] Ludwig Merkle. *Bairische Grammatik*. Hugendubel, 1975. 18, 20, 28, 29, 30

[sam04] Sampa - computer readable phonetic alphabet. Online, 2004. http://www.phon.ucl.ac.uk/home/sampa/home.htm, zuletzt gesichtet am 22. Juni 2004.

[Ste94] Reinhold Steininger. *Beiträge zu einer Grammatik des Bairischen*. Franz Steiner Verlag Stuttgart, 1994.

[tm00] The MBROLA team (mbrola@tcts.fpms.ac.be). instructions-for-diphones. Textdatei, 2000. http://tcts.fpms.ac.be/synthesis/mbrola/docs/mbrola_instructions.zip. 40, 44

[Zeh85] Ludwig Zehetner. *Das bairische Dialektbuch*. Verlag C. H. Beck, 1985. 23, 29, 31