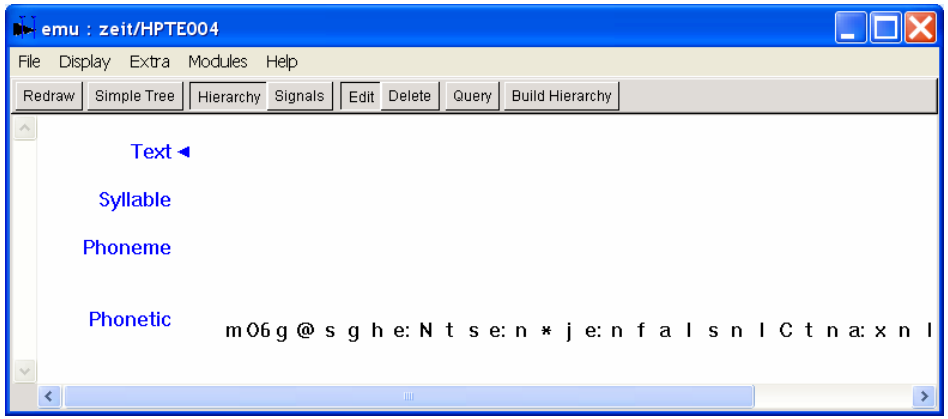
Anwendung von Emu-TCL

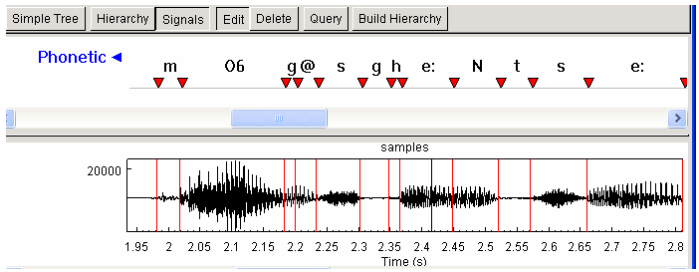
Jonathan Harrington

Emu-TCL: Ein Interface zwischen Emu und der Programmiersprache TCL.

Mit diesem Interface und TCL-Funktionen können Etikettierungsstrukturen teilweise automatisch erstellt werden.

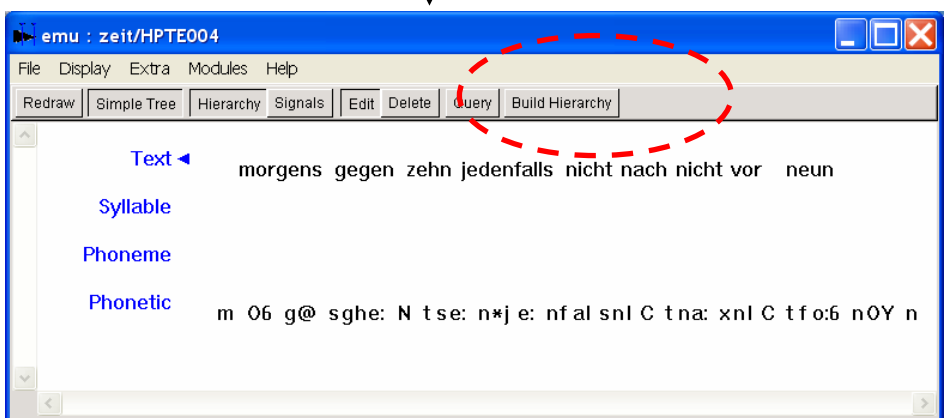
Ziel 1: Text (Orthographie) in eine Ebene einlesen



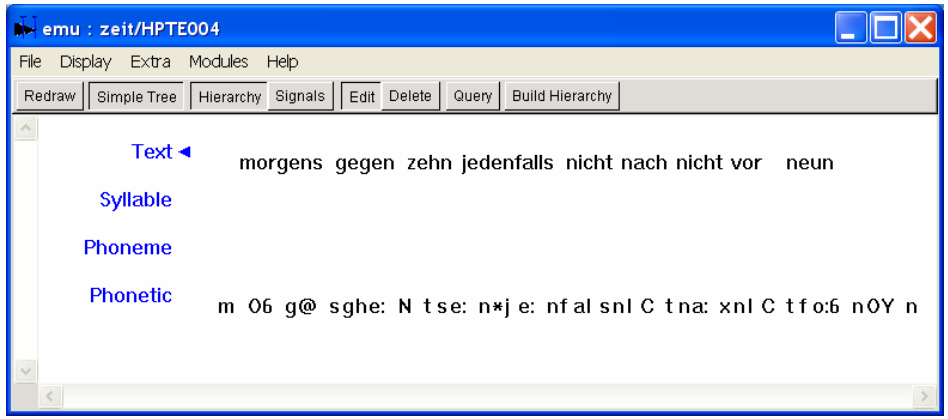


AddLabelsFromFile

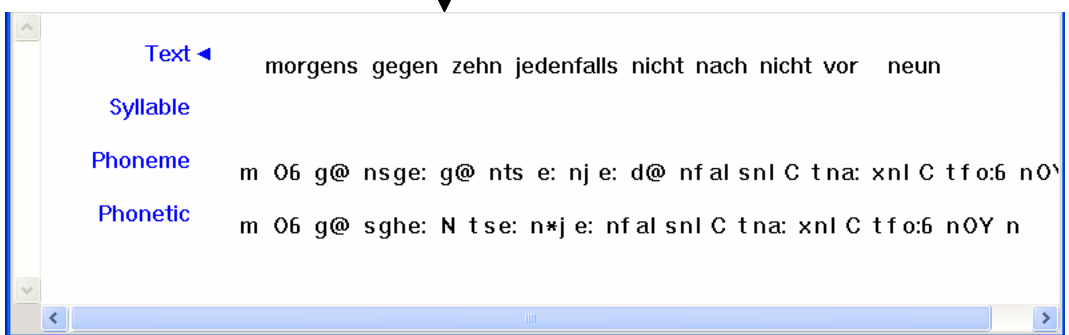
Text-Datei der Orthographie



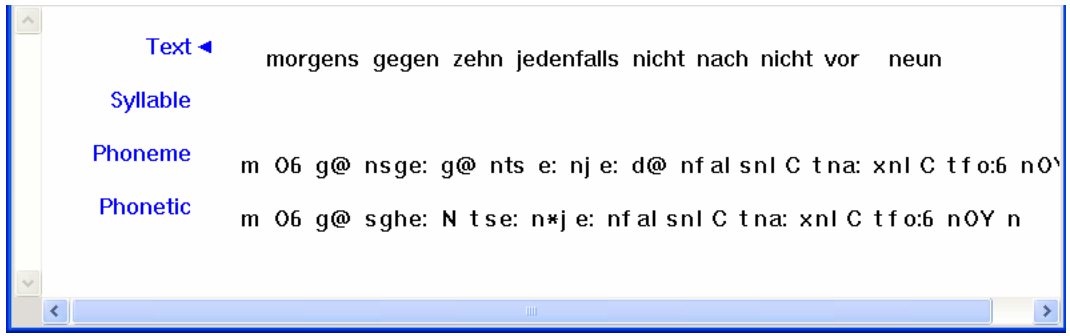
Ziel 2 Phoneme der Wörter aus einem Lexikon einlesen

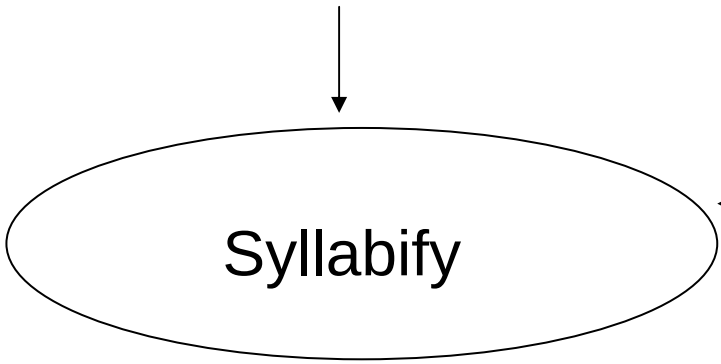


Text-Datei
eines Lexikons

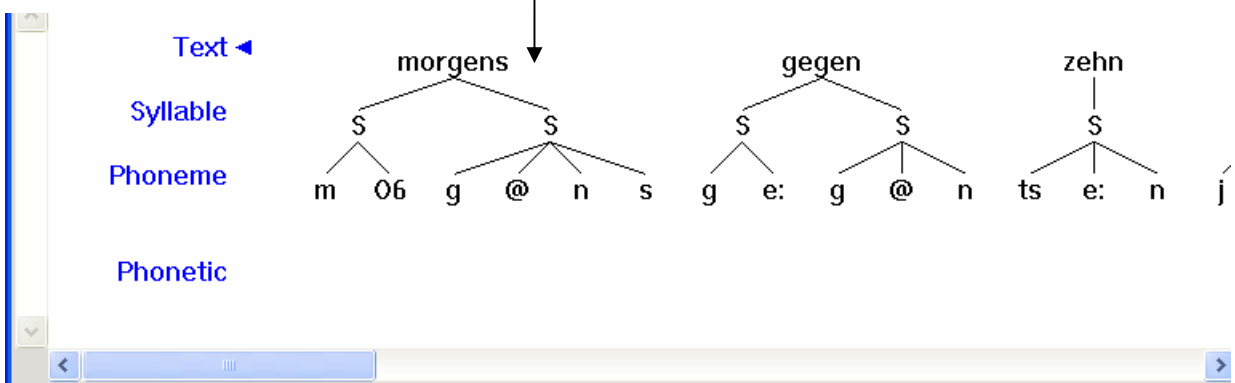


Ziel 3: Aufbau einer Silbenstruktur





Text-Datei der erlaubten Ks, die eine Silbe beginnen dürfen



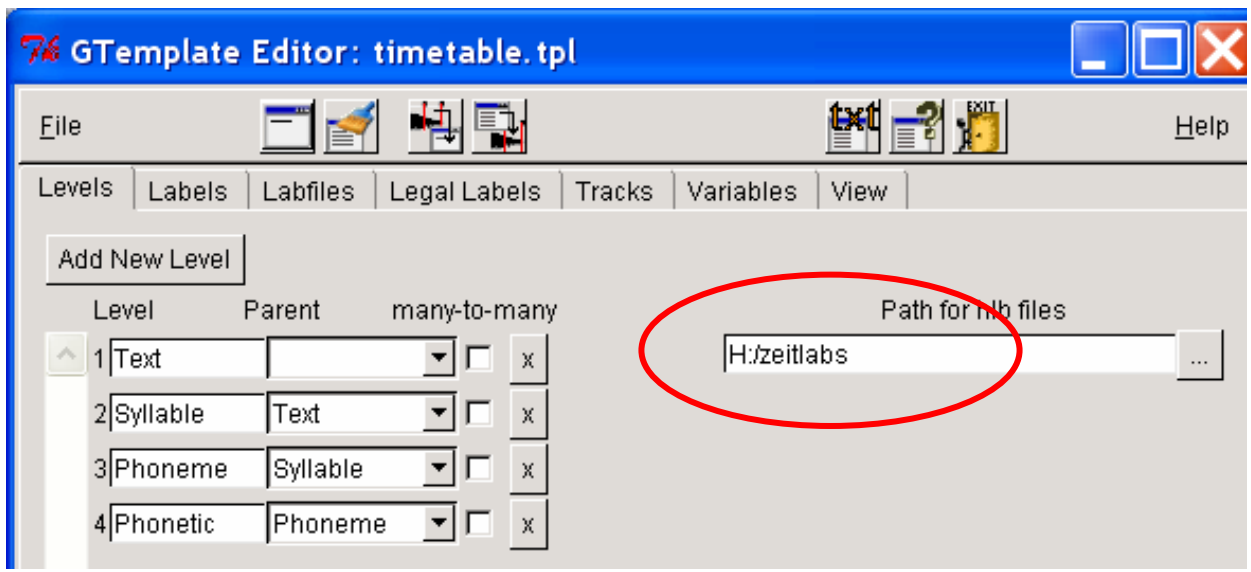
Vorverarbeitung: Erstellung einer Template-Datei

1. Ein Verzeichnis erzeugen, in dem die hlb-Dateien gespeichert werden – z.B. eigene dateien\zeitlabs

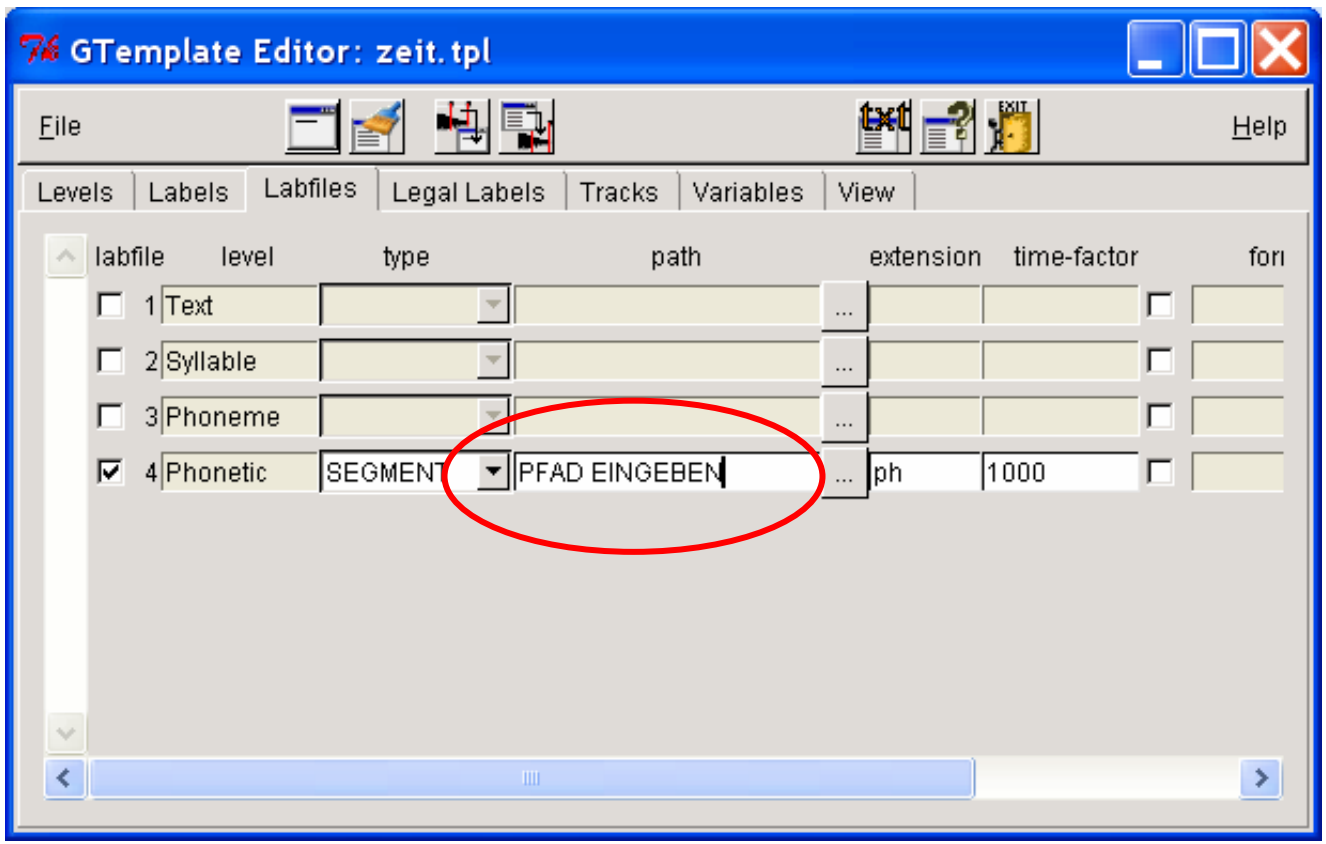
2. Die Text-Datei S:/IPSK/EMUKoll/dbs/timetable/emutcl.txt nach diesem Verzeichnis kopieren

3. Die Template-Datei **timetable** editieren

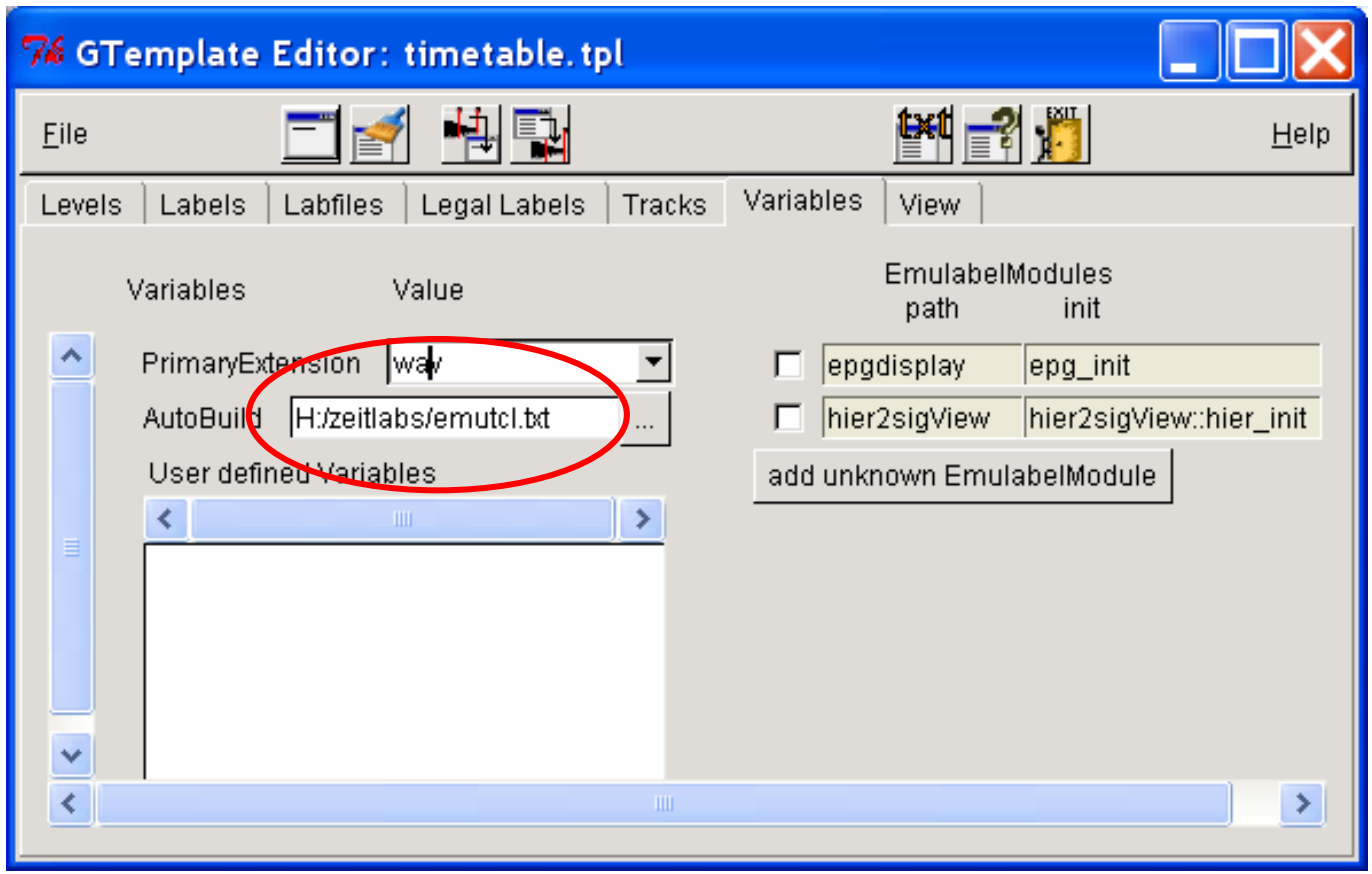
4. Den Pfad aus 1. für die hlb-Dateien definieren.



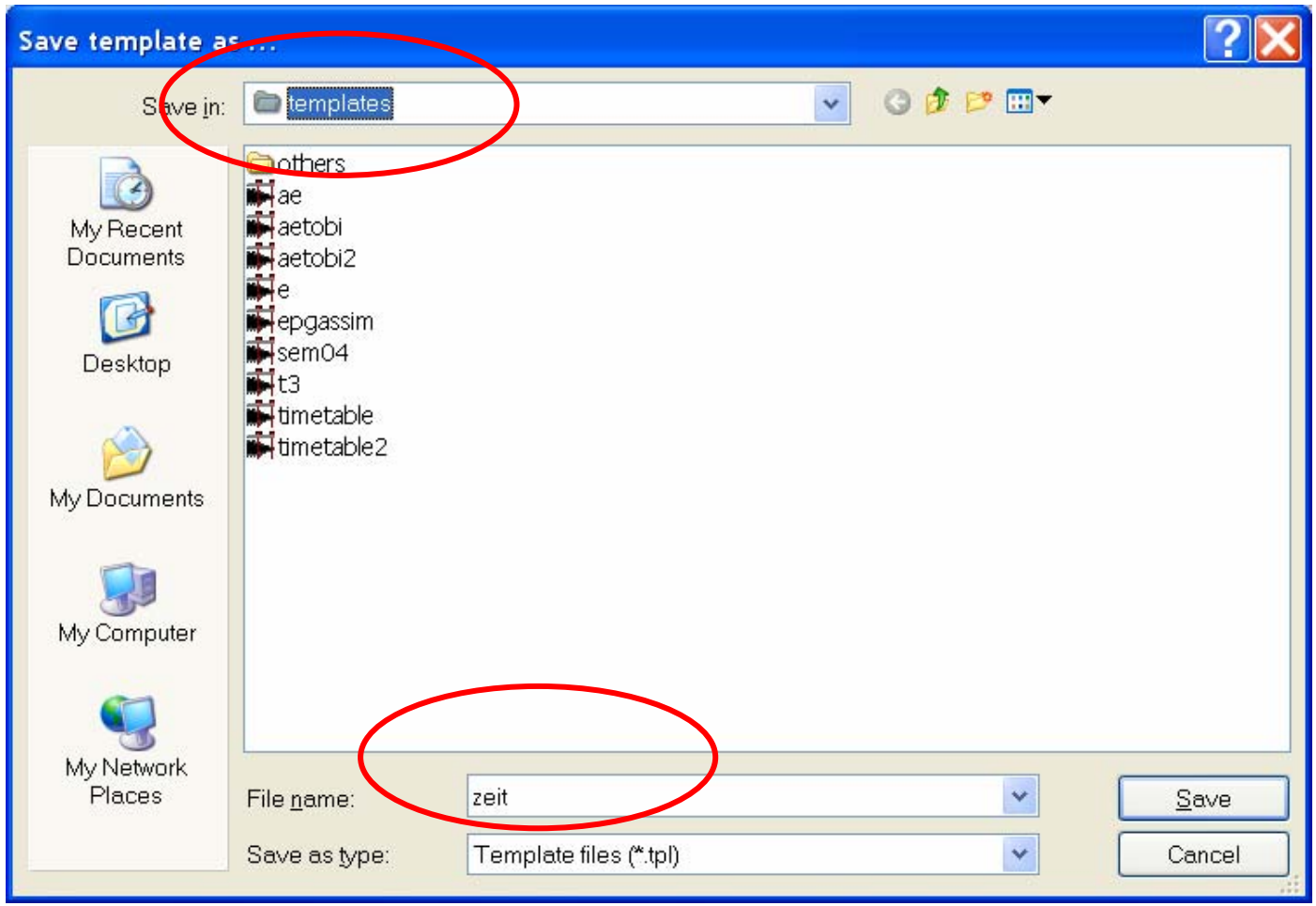
5. Den Pfad für die phonetischen Etikettierungen eingeben. Dies soll derselbe Pfad sein, den Sie vorige Woche in der Übung verwendet haben (= Verzeichnis, in dem Sie HPTE001.ph, HPTE002.ph…HPTE005.ph gespeichert haben)



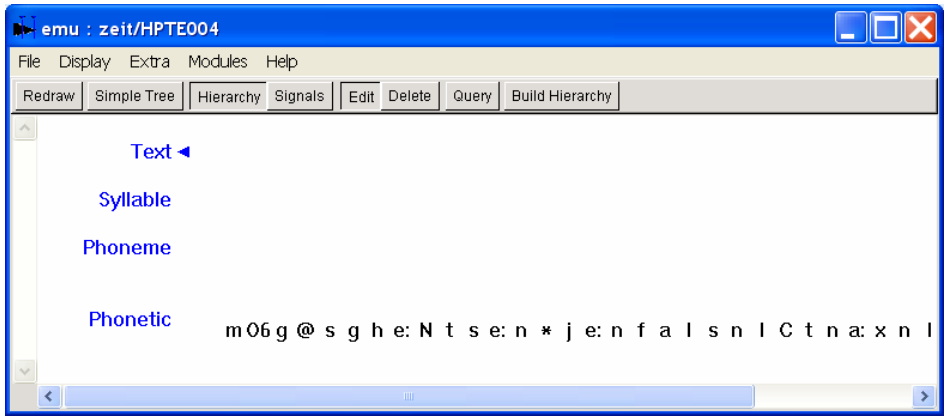
6. Die emutcl.txt Datei einlesen:



7. Die Template-Datei **mit einem neuen Namen in Ihrem Template-Verzeichnis** speichern

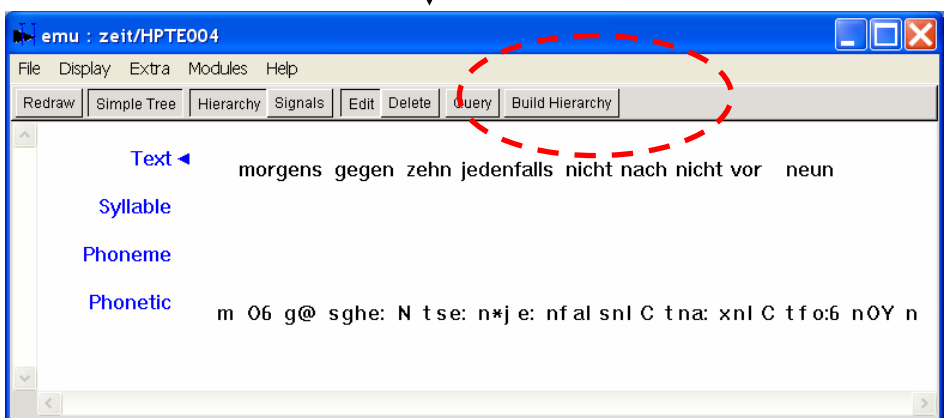


Erste Aufgabe



AddLabelsFromFile

Text-Datei der Orthographie



Die Datei emutcl.txt editieren

```
package require emu::autobuild

proc AutoBuildInit {template} {
}


proc AutoBuild {template tree} {
}
```

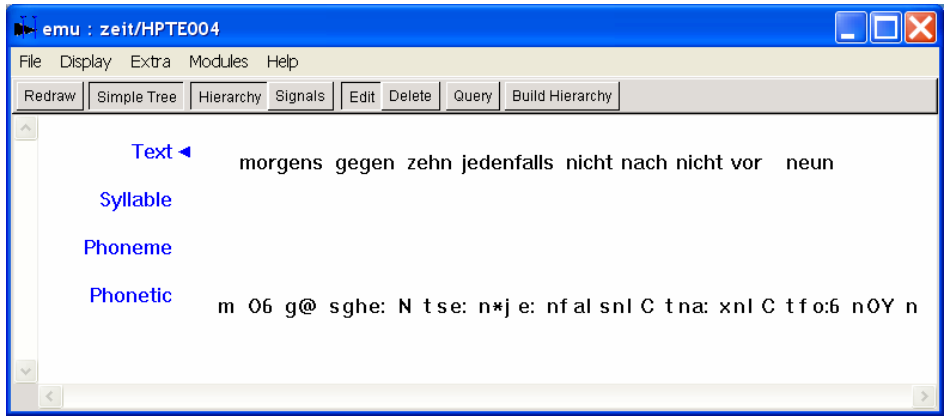
Bereits vorhanden.

Kommentare (fakultativ)

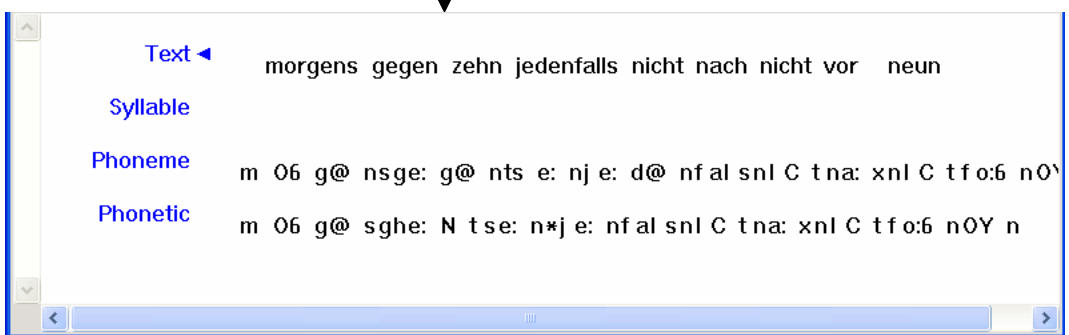
Neue Befehle

```
package require emu::autobuild
proc AutoBuildInit {template} {

}

proc AutoBuild {template tree} {
# in welchem Verzeichnis sind die Text-Dateien?
set wordpath S:/IPSK/EMUKoll/dbs/timetable/orthography
# Funktion ausführen
AddLabelsFromFile $template $tree $wordpath Text
}
```

Ziel 2 Phoneme der Wörter aus einem Lexikon einlesen

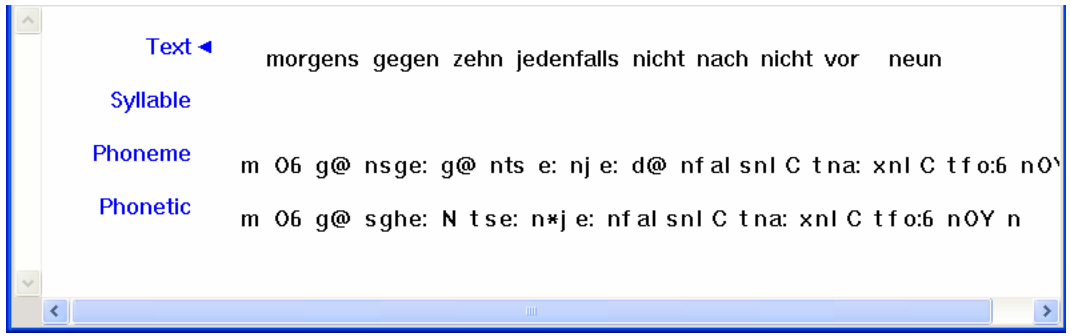


Text-Datei
eines Lexikons



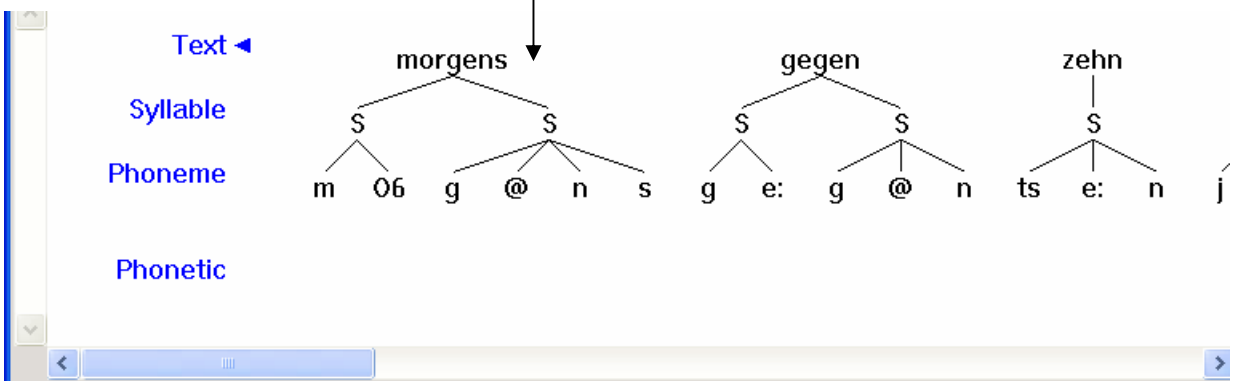
```
package require emu::autobuild
proc AutoBuildInit {template} {
}

proc AutoBuild {template tree} {
set wordpath S:/IPSK/EMUKoll/dbs/timetable/orthography
AddLabelsFromFile $template $tree $wordpath Text

# Lexikon einlesen
InitialiseDict lex S:/IPSK/EMUKoll/dbs/timetable/dictgerman.txt
# Befehl ausführen: Phoneme aus dem Lexikon einlesen
LevelFromDict $tree Text Phoneme lex
}
```

Ziel 3: Aufbau einer Silbenstruktur



Syllabify

Text-Datei der erlaubten Ks, die eine Silbe beginnen dürfen



Silbifizierung nach dem Maximum-Onset-Prinzip vornehmen

(so viele Ks mit einem folgendem Vokal wie möglich silbifizieren, vorausgesetzt dass die die K-Reihenfolge phonotaktisch legal ist)

zB 'alter' = al . ter

```
package require emu::autobuild
proc AutoBuildInit {template} {
}

proc AutoBuild {template tree} {
set wordpath S:/IPSK/EMUKoll/dbs/timetable/orthography
AddLabelsFromFile $template $tree $wordpath Text

InitialiseDict lex S:/IPSK/EMUKoll/dbs/timetable/dictgerman.txt
LevelFromDict $tree Text Phoneme lex

# die legalen, silbeninitialen Ks einlesen
source S:/IPSK/EMUKoll/dbs/timetable/clusters.txt
# Die Silbifizierung durchfuehren
Syllabify $template $tree Phoneme Syllable cons
}
```

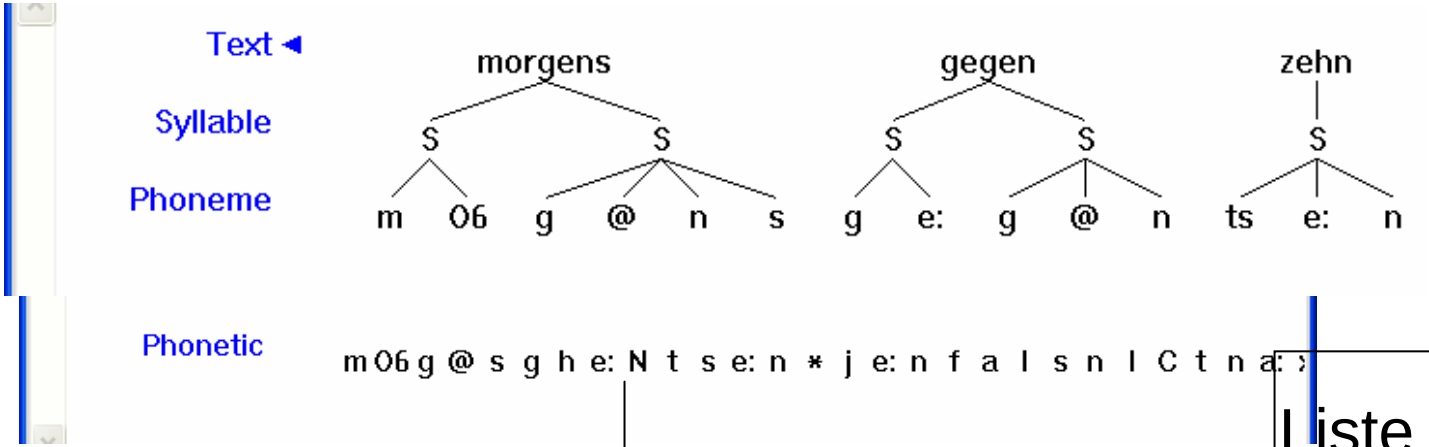
Phoneme und Phonetic verbinden

Diese Ebenen unterscheiden sich hauptsächlich aufgrund von Verschleifungen, Reduzierungen usw.

Diese Ebenen werden automatisch miteinander verbunden anhand der EMU-TCL Funktion **InsertWordBoundaries**, die zwei ähnliche, jedoch unterschiedliche, Etikettierungen auf eine optimale Weise miteinander zu verbinden versucht.

Beispiele der Verschleifungen in diesem Korpus

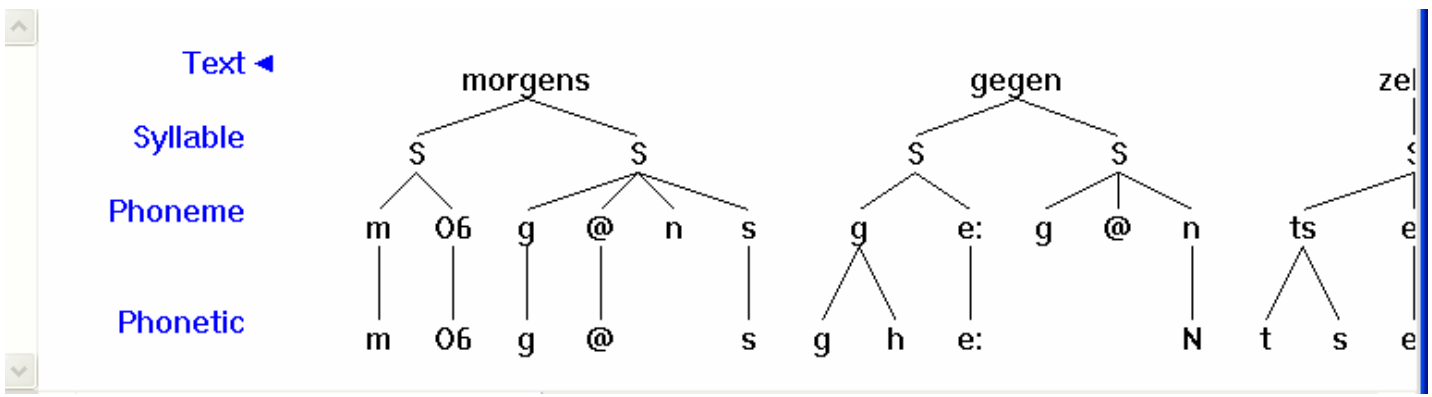
Utterance	**Lexical entry**	**Phoneme (Lexikon)**	**Phonetic**	**gloss**
HPTE001	morgen	mO6g@n = /mɔɐɡən/	mo6gN = [mɔɐɡŋ]	tomorrow
HPTE002	morgens	mO6g@ns = /mɔɐɡəns/	mO6Ns = [mɔɐŋs]	in the morning
HPTE004	gegen	ge:g@n = /ɡe:ɡən/	ghe:N = [ɡˣe: ŋ]	about
HPTE004	jedenfalls	je:d@nfals = /je:dənfals/	je:nfals = [je: nfals]	in any case
HPTE005	möglichst	m2:klICst = /mø:klɪçst/	m2:klIC = [mø:klɪç]	as possible
HPTE005	Abend	a:b@nt = /a:bənt/	a:bn = [a:bn]	evening



InitialiseDict

InsertWordBoundaries

Liste der Default Beziehungen zwischen phonemischen und phonetischen Einheiten



Beispiel der Default-Beziehungen zwischen Phoneme und Phonetic

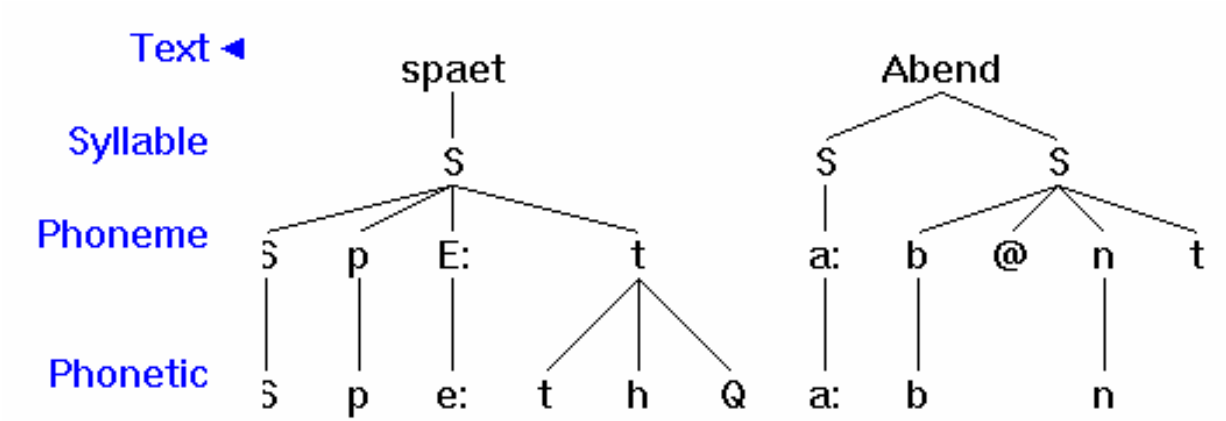
d d
k k
g g
pf p f
ts t s
h h

S:\IPSK\EMUKoll\dbs\timetable\phonemesgerman.txt

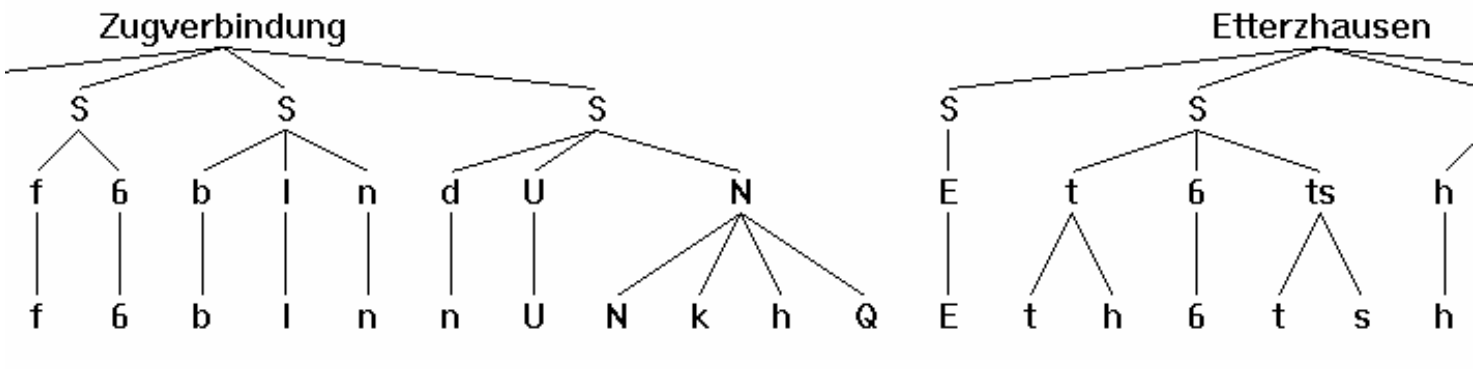
```
package require emu::autobuild
proc AutoBuildInit {template} {
}

proc AutoBuild {template tree} {
set wordpath S:/IPSK/EMUKoll/dbs/timetable/orthography
AddLabelsFromFile $template $tree $wordpath Text

InitialiseDict lex S:/IPSK/EMUKoll/dbs/timetable/dictgerman.txt
LevelFromDict $tree Text Phoneme lex

source S:/IPSK/EMUKoll/dbs/timetable/clusters.txt
Syllabify $template $tree Phoneme Syllable cons

# eine Liste der deutschen Phoneme und deren 'Default' Beziehungen zu
# phonetischen Einheiten
InitialiseDict phonemes S:/IPSK/EMUKoll/dbs/timetable/phonemesgerman.txt
# die best moegliche Assoziation zwischen Phonetic- und Phoneme-Ebenen
InsertWordBoundaries $template $tree phonemes Phoneme Phonetic
}
```

Zwei Beispiele von Fehlern

Glottalverschluss mit /t/ statt /A/

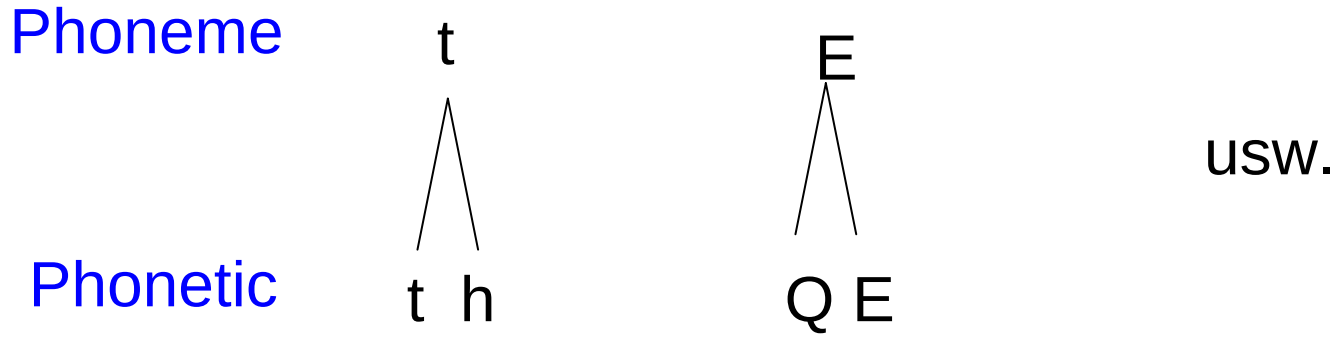


/N/ aus 4 phonetischen Einheiten

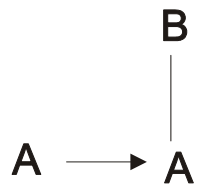
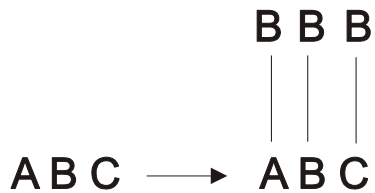
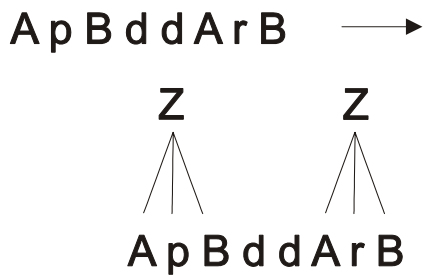


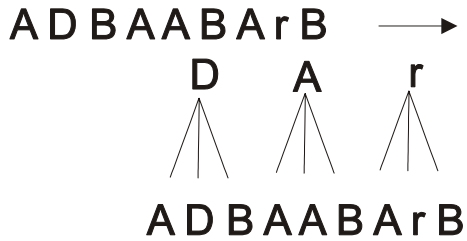
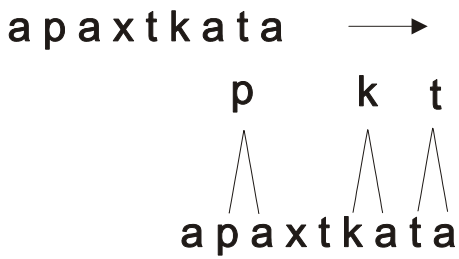
Mit der **MapLevels** Funktion können wir Regeln anwenden um die allophonische Variation zu reduzieren

Die Regeln sind 'bottom-up' d.h. sie erstellen Verbindungen von der unter geordneten Phonetic- zur uebergeordneten Phoneme-Ebene



Einige Beispiele der möglichen Regeln mit der MapLevels Funktion

A -> B	Create B and link it to A	
t S -> tS	Create tS and link it to t S	
<x> -> B	Create as many B's as there are labels and link each one separately	
A <x> B -> Z	Create Z and link it to a sequence of A, any label, B	

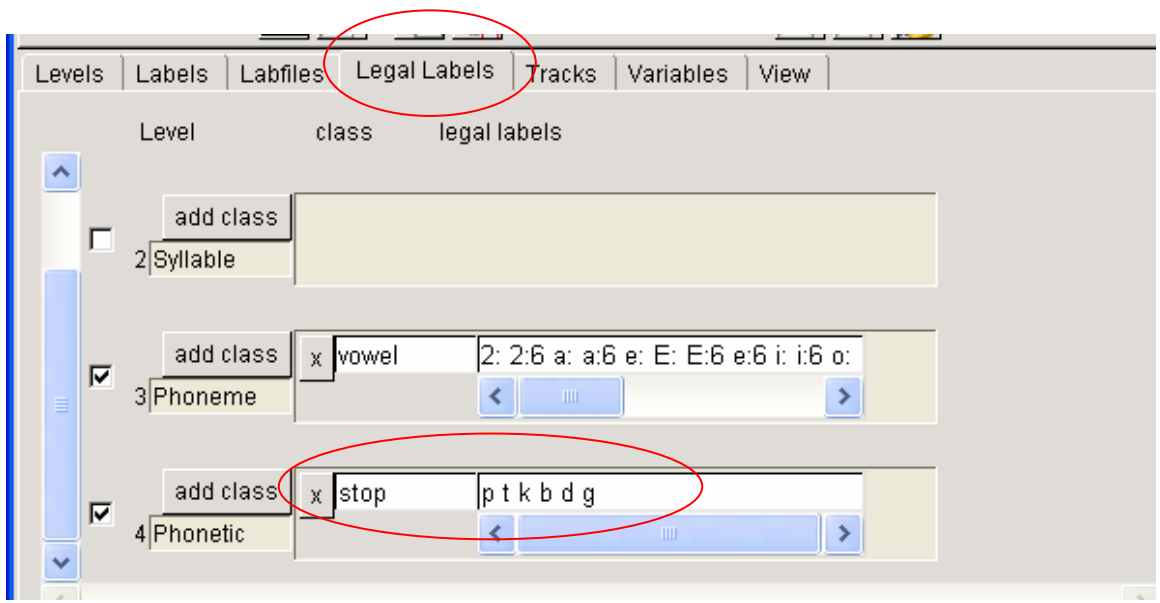
A <x> B -> <x>	Link a sequence of A, any label, B under whatever label is between A and B	
A -> A B	Create A B and link them to A (requires the tiers to be in a many-to-many assocation)	
<p=stop> a -> <p>	Create and link segments defined by stop at both levels (requires a feature definition of “stop” in the template file)	

Die Regeln, die wir anwenden werden

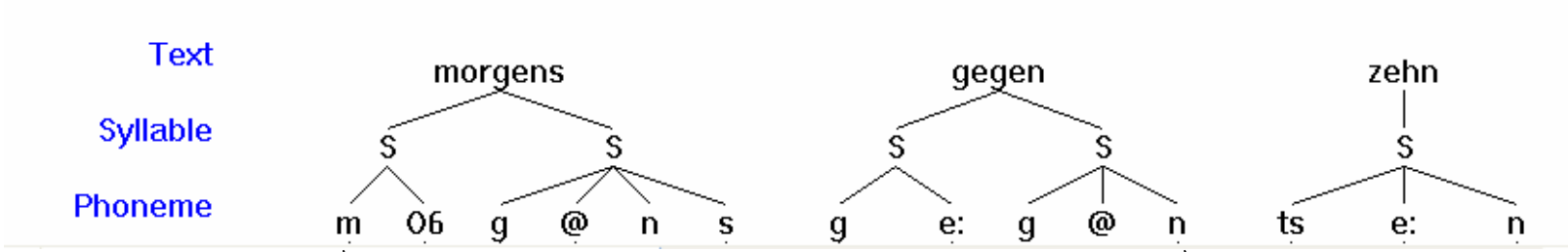
siehe S:\IPSK\EMUKoll\dbs\timetable\prules.txt

Q <x> -> <x>
<x=stop> h -> <x>
<x> -> <x>

Regel 2 setzt voraus, dass **stop** als Merkmal in der Template-Datei definiert worden ist

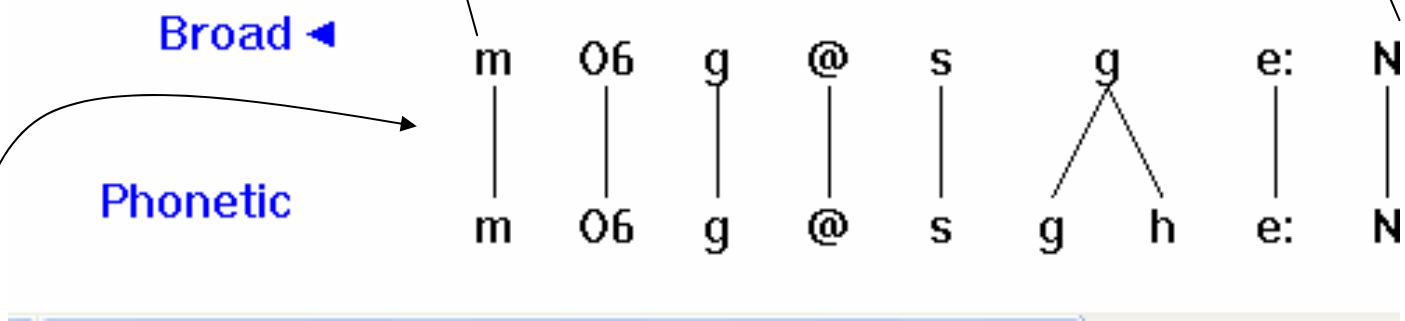


Der Vorgang



1.Neue Ebene 'Broad' definieren

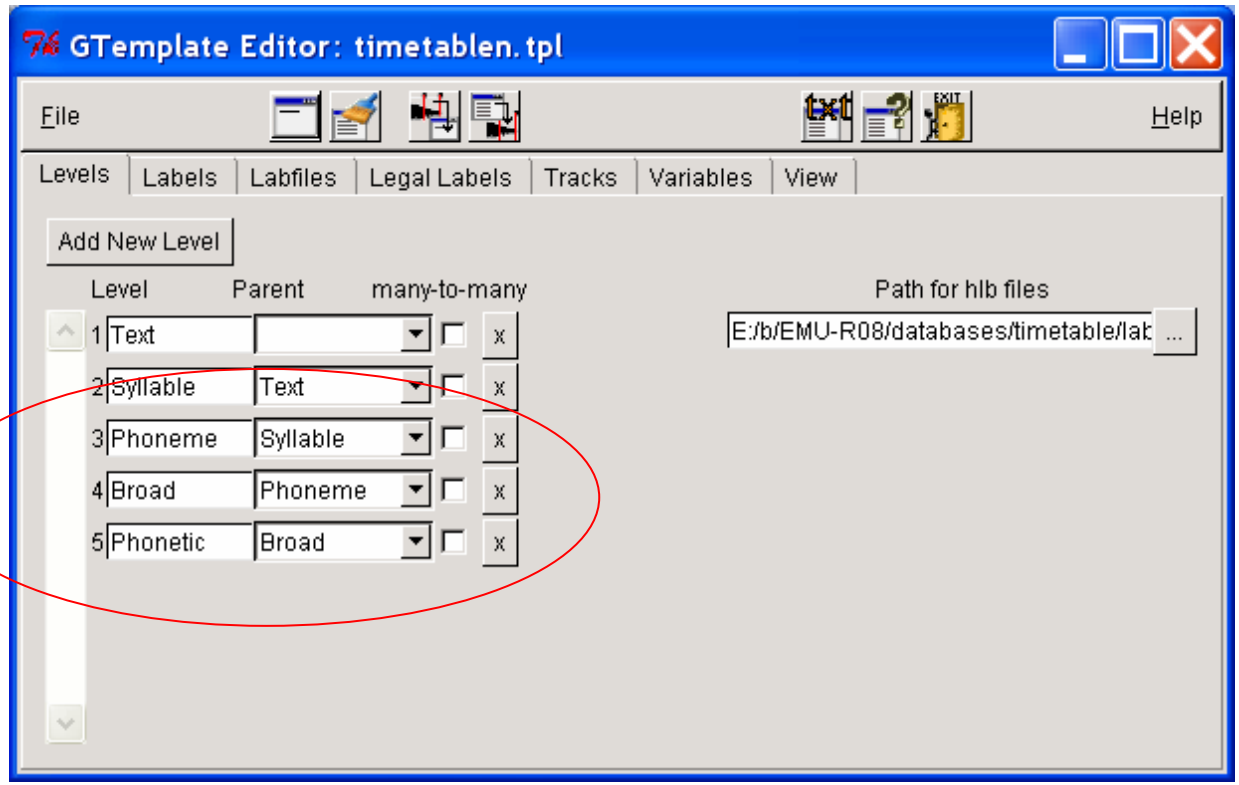
3. InsertWordBoundaries



2. ReadLevelRules, MapLevels

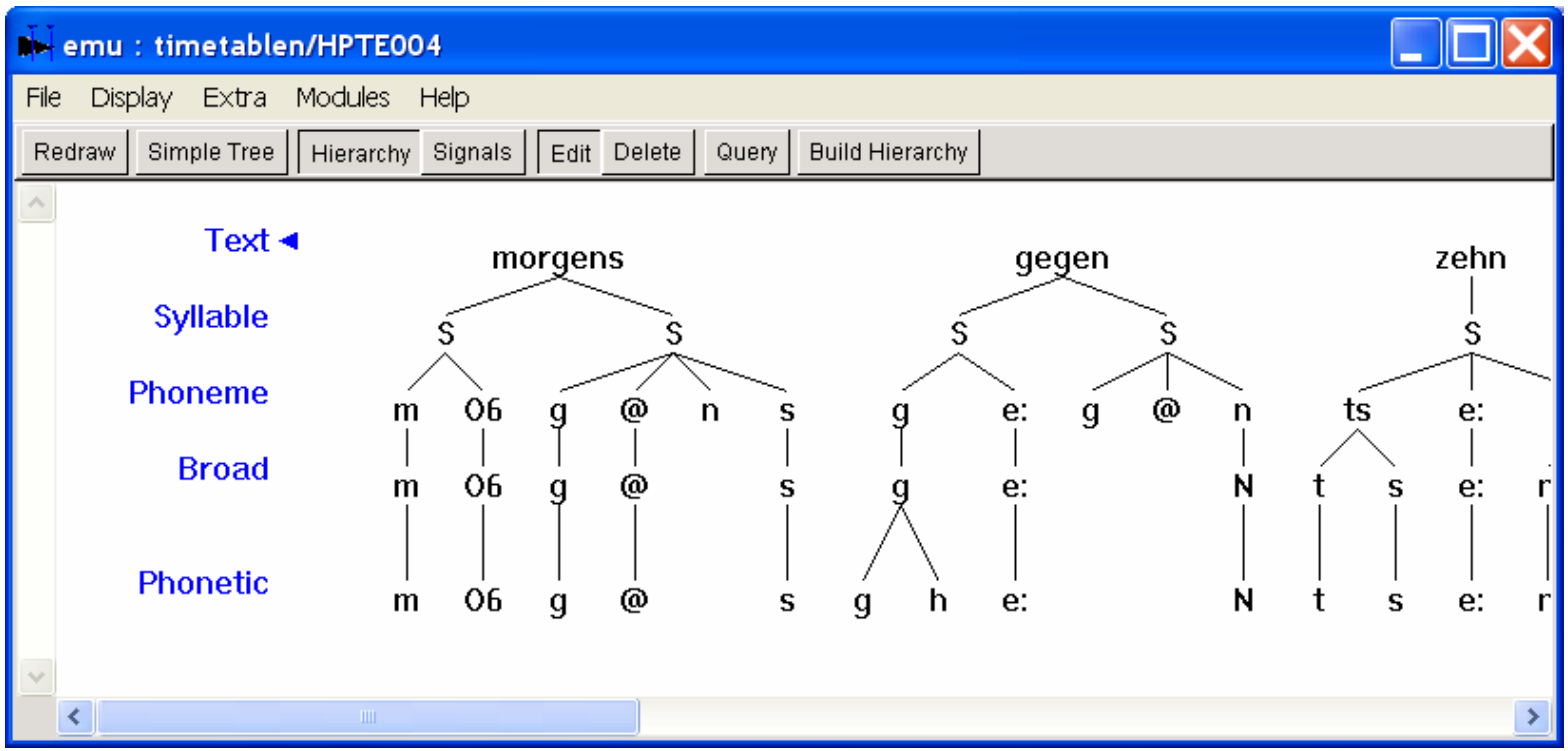
Die Phonetic-Broad Regeln

Template-Datei ändern



```
package require emu::autobuild
proc AutoBuildInit {template} {
}

proc AutoBuild {template tree} {
set wordpath S:/IPSK/EMUKoll/dbs/timetable/orthography
AddLabelsFromFile $template $tree $wordpath Text

InitialiseDict lex S:/IPSK/EMUKoll/dbs/timetable/dictgerman.txt
LevelFromDict $tree Text Phoneme lex

source S:/IPSK/EMUKoll/dbs/timetable/clusters.txt
Syllabify $template $tree Phoneme Syllable cons

# Regeln einlesen
set prules [ReadLevelRules S:/IPSK/EMUKoll/dbs/timetable/prules.txt]
# Die Ebenen Broad und Phonetic miteinander verbinden
MapLevels $template $tree Broad Phonetic $prules

InitialiseDict phonemes S:/IPSK/EMUKoll/dbs/timetable/phonemesgerman.txt
# Phoneme mit Broad verbinden
InsertWordBoundaries $template $tree phonemes Phoneme Broad
}
```

Das Ergebnis



Die EMU-TCL Funktion kann auf alle Äußerungen auf einmal mit Database operations -> AutoBuild extern angewandt werden.