AutoBuild Prozeduren im package emu::autobuild

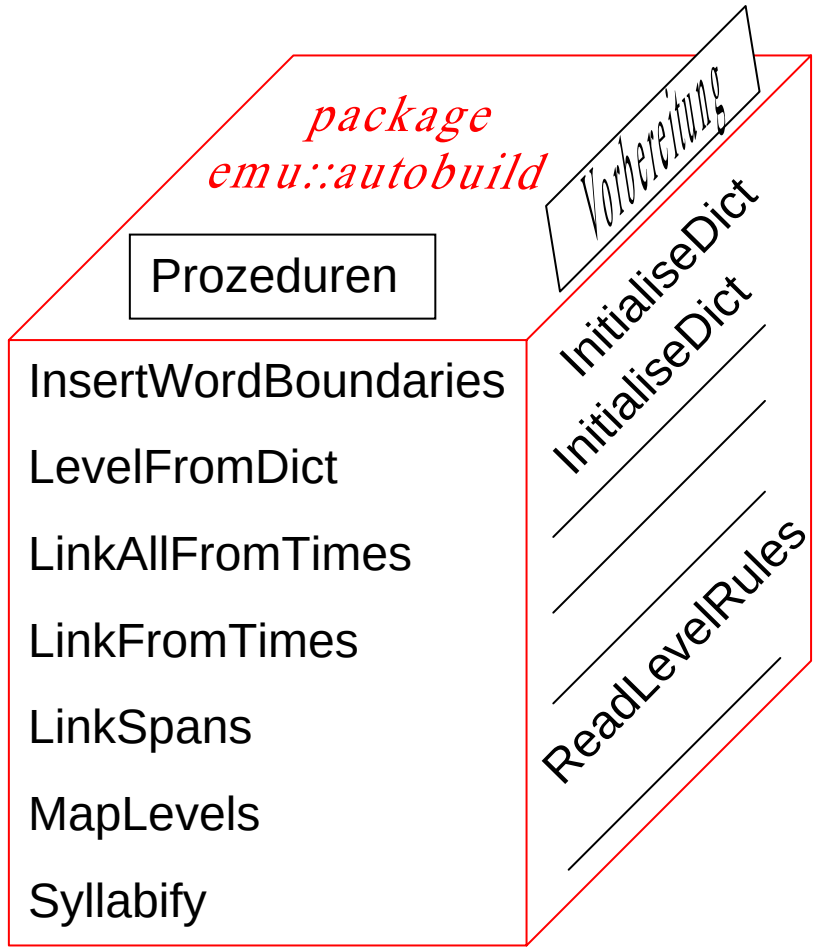
Tina John

AutoBuild Prozeduren im package emu::autobuild

autobuild_default.tcl - Editor

Datei Bearbeiten Format Ansicht ?

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {

}
```

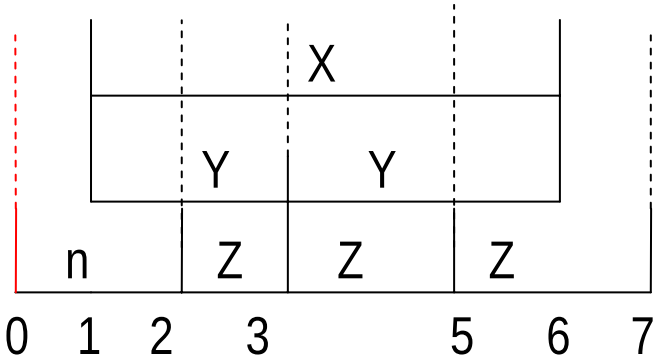


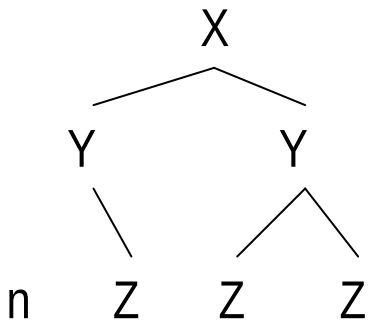
Prozedur mit Parametern	**In proc AutoBuildInit {templ} {…}**	**Form der zusätzlichen txt Datei**	**In proc AutoBuild {templ hier} {…}**
proc InsertWordBoundaries {templ hier dict plevel clevel}	InitialiseDict dictionary dict.txt	die d *i: sonne s *O n @ lacht l *a x t	InsertWordBoundaries $templ $hier dictionary Word Canonic
proc LevelFromDict { hier parent child dictionary}	InitialiseDict dictionary dict.txt	die d *i: sonne s *O n @ lacht l *a x t	LevelFromDict $hier Word Phoneme dictionary
proc LinkAllFromTimes {templ hier}			LinkAllFromTimes $templ $hier
proc LinkFromTimes {hier plevel clevel}			LinkFromTimes $hier Word Tone
proc LinkSpans {hierarchy plevel clevel}			LinkSpans $hier Word Phonetic
proc MapLevels {templ hier plevel clevel rules}	set rules [ReadLevelRules rules.txt]	<s=stop> h -> <s> t s -> ts p f -> pf	MapLevels $templ $hier Kanonik Phonetik $rules
proc Syllabify {tpl hier plevel slevel goodarray}		In der TemplateDatei müssen legal labels definiert sein	set cons(triples) {pfl pfr tsv Spr Spl Str skr skl } set cons(pairs) {pr br tr dr kr gr fr Sr pl bl kl gl fl Sl Sm kn gn Sn kv Sv Sp St sk} set cons(singles) {p b t d k g pf ts dZ f v s z S Z C x m n N l r j h} Syllabify $templ $hier Canonic Syllable cons

LinkAllFromTimes

Prozeduraufruf mit Parametern

LinkAllFromTimes $templ $hier





Funktion

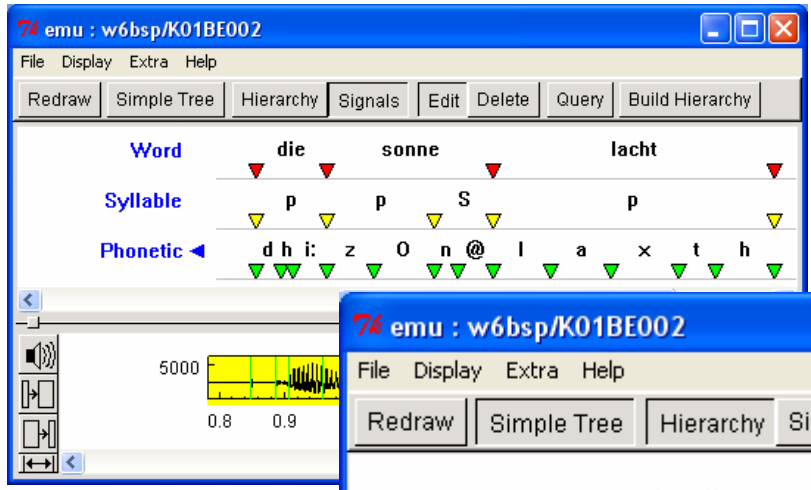
Setzt Links zwischen Segmenten auf allen zeitgebundenen Ebenen, die miteinander assoziiert sind.

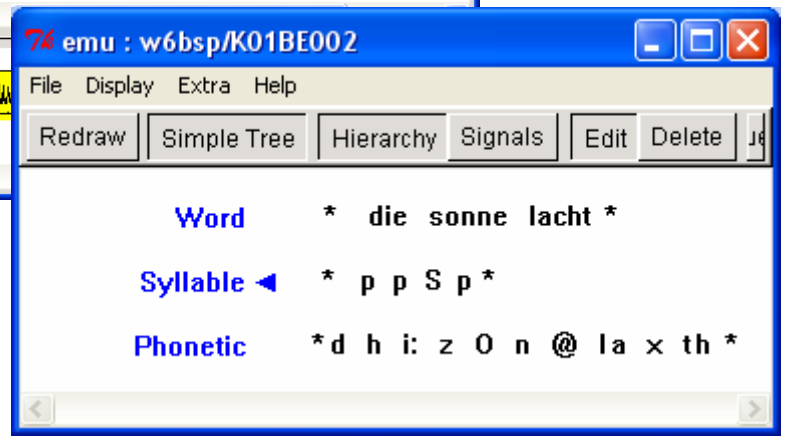
Das Kriterium für die Verknüpfung ist die Zeit.

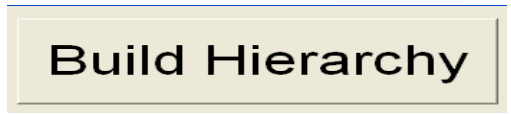
Sobald die Onset-Marke des Segments auf der Kind-Ebene innerhalb der Grenzen des Segmentes auf der Eltern-Ebene liegt, werden diese beiden Segmente verbunden.

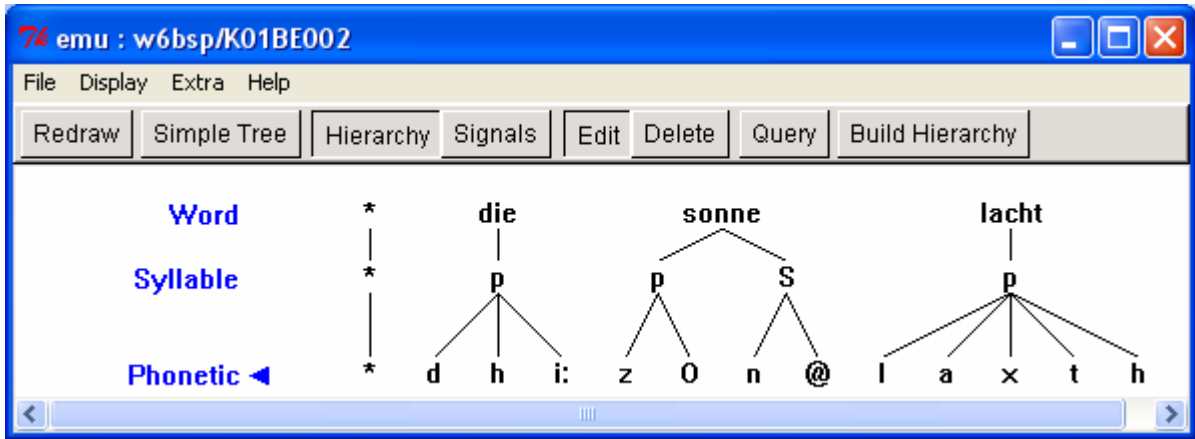
LinkAllFromTimes

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {

  LinkAllFromTimes $templ $hier
}
```









LinkFromTimes

Prozeduraufruf mit Parametern

LinkFromTimes $hier plevel clevel

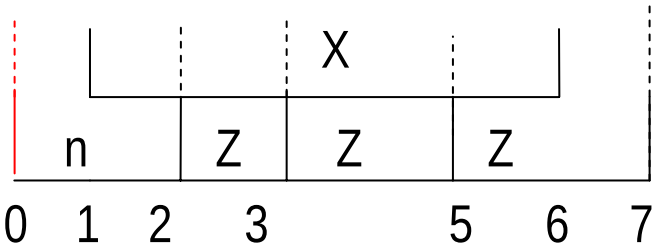
plevel = die dominierende Ebene

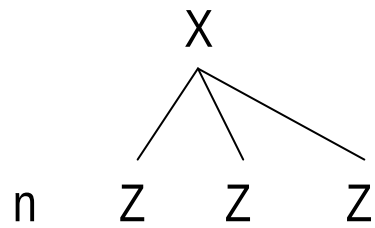
clevel = die von plevel dominierte Ebene

Funktion

Verbindet Segmente auf zwei assoziierten zeitgebundenen Ebenen.

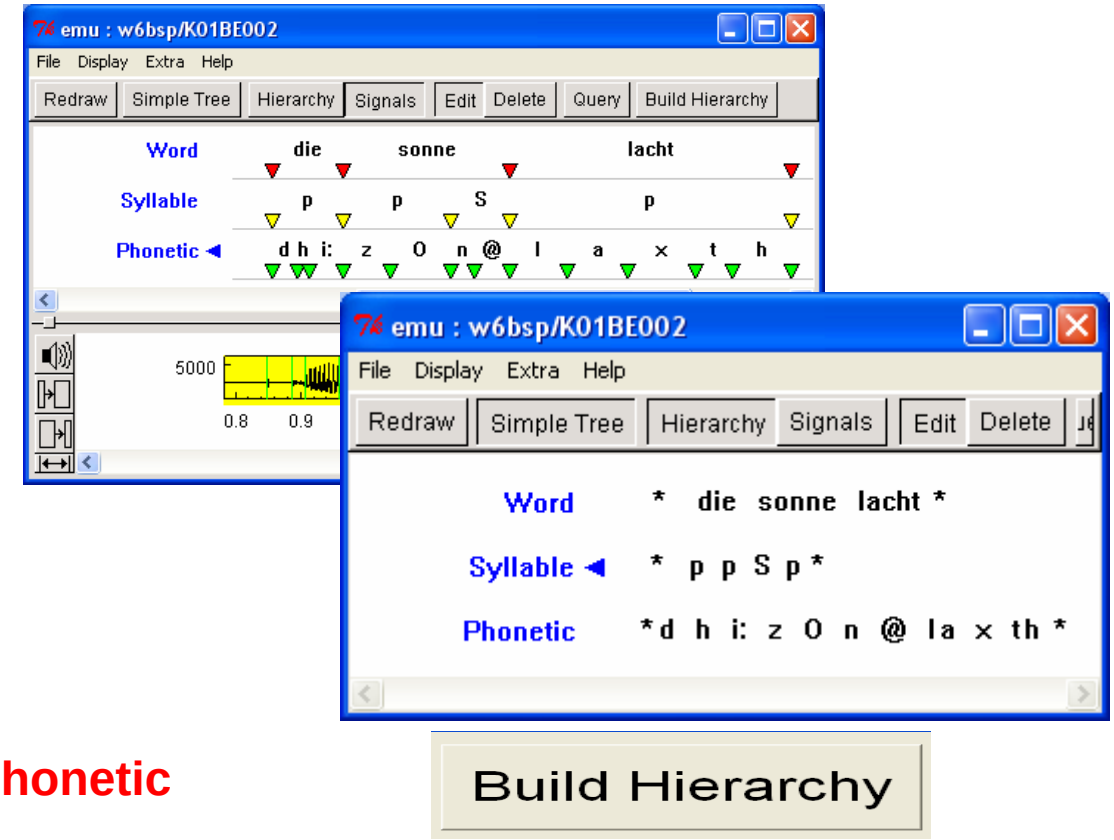
Sobald die Onset-Marke des Segments auf der Kindebene innerhalb der Grenzen des Segmentes auf der Eltern-Ebene liegt, werden diese beiden Segmente verbunden.

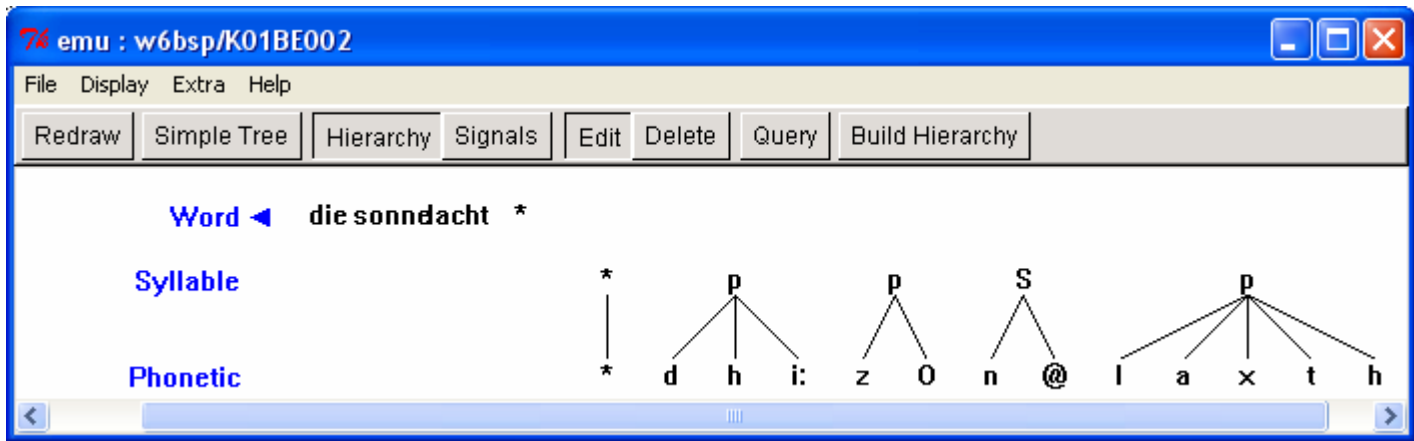




LinkFromTimes

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {

  LinkFromTimes $hier Syllable Phonetic
}
```





LinkSpans

Prozeduraufruf mit Parametern

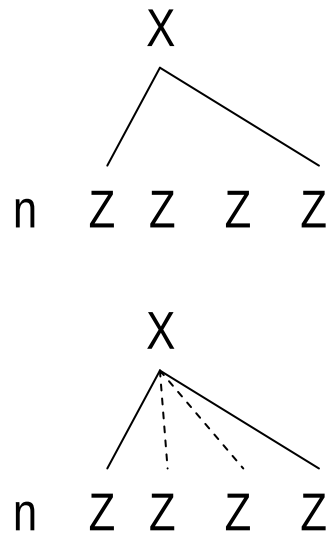
LinkSpans $hier plevel clevel

plevel = die dominierende Ebene

clevel = die von plevel dominierte Ebene

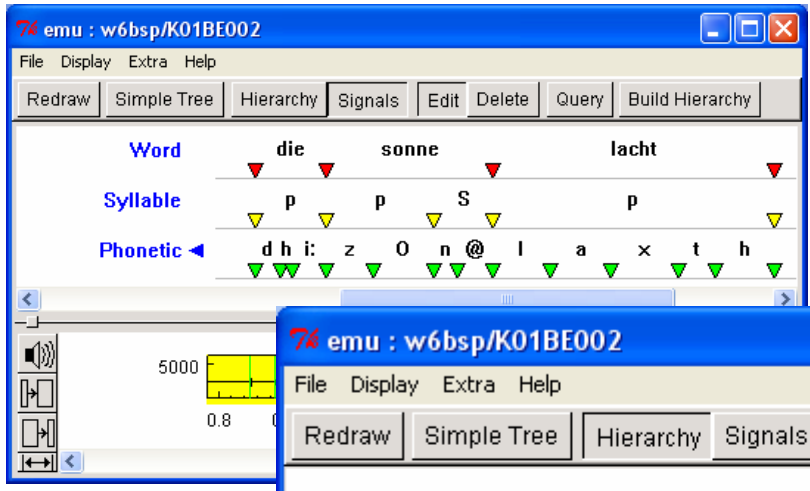
Funktion

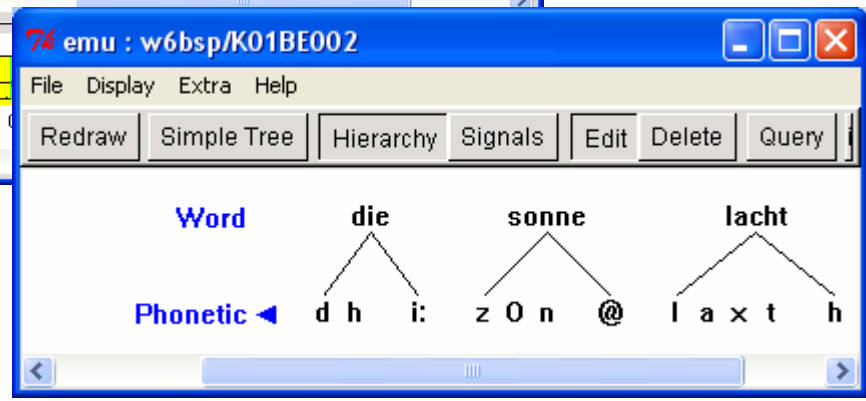
Segmente auf einer Kind-Ebene werden mit einem Segment auf der Elternebene verbunden, wenn sich diese Segmente zwischen 2 Segmenten auf der Kind-Ebene befinden, die bereits mit dem Segment auf der Eltern-Ebene verbunden sind.

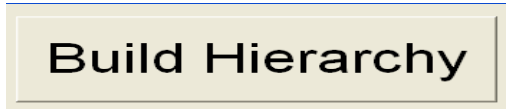


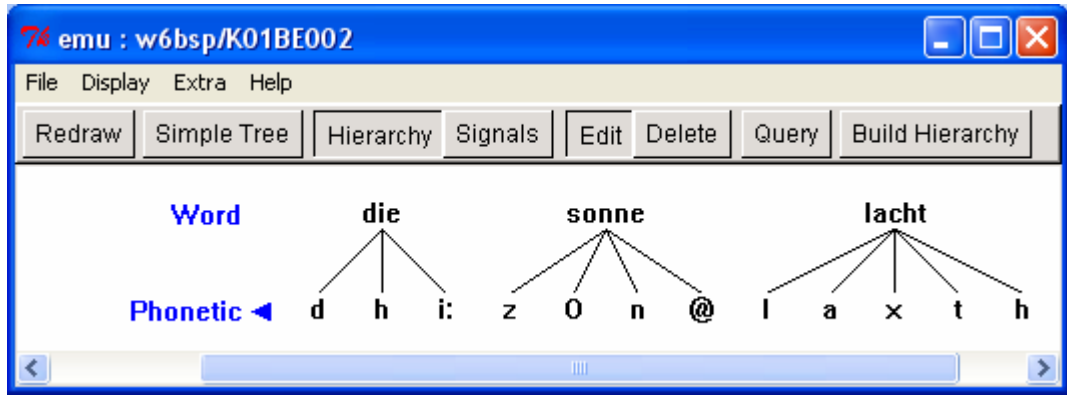
LinkSpans

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {

  LinkSpans $hier Word Phonetic
}
```









InsertWordBoundaries

Prozeduraufruf mit Parametern

InsertWordBoundaries $templ $hier dictionary plevel clevel

plevel = die dominierende Ebene

clevel = die von plevel dominierte Ebene

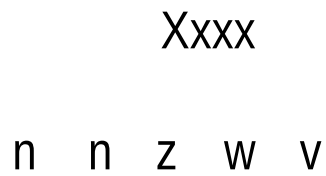
Benötigt als Vorbereitung die Prozedur **InitialiseDict**

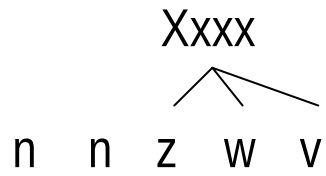
Funktion

Verbindet bereits vorhandene Segmente auf zwei assoziierten Ebenen.
Kriterien für das Verbinden sind in einem dictionary festgehaltene (selbsteditiert) Regeln.

Dictionary

Xxxx z w v





InsertWordBoundaries InitialiseDict

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global dictionary
InitialiseDict dictionary dict.txt

}

proc AutoBuild {templ hier} {
global dictionary
  InsertWordBoundaries $templ $hier dictionary
Word Canonic

}
```

dict.txt - Editor

Datei Bearbeiten Format Ansicht ?

```
die  d  i:
sonne s O n @
lacht l a x t
heute h OY t @
ist I s t
schoenes S 2: n @ s
fruehlings f r y: l I N s
wetter v e t @
am  a m
blauen b l au n
himmel h I m @ l
ziehen ts i: @ n
wolken v O l k @ n
```

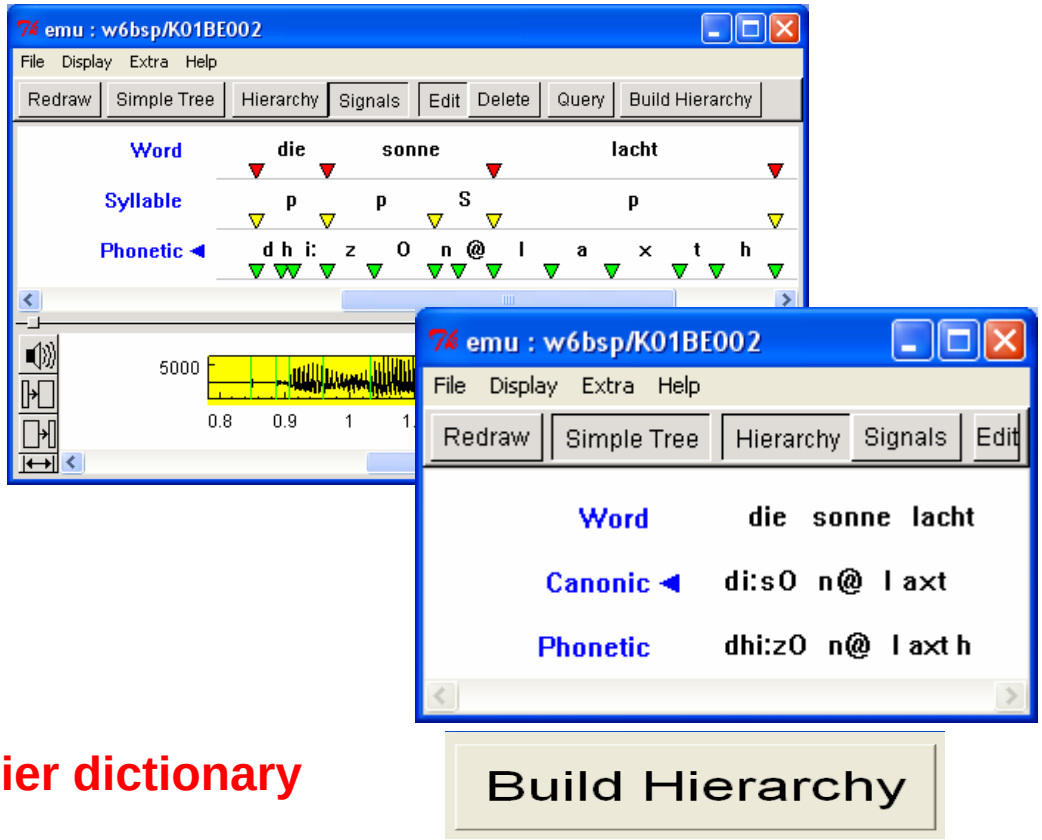
Die vorhandenen Segmente auf der Eltern-Ebene

Die vorhandenen Segmente auf der Kind-Ebene

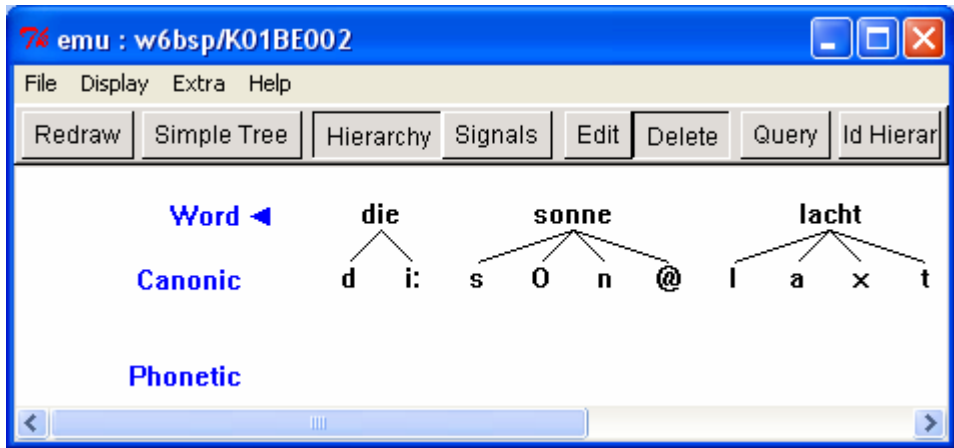
Es müssen alle vorkommenden Segmente auf der Eltern und Kind-Ebene im dictionary definiert sein. Leere Segmente müssen auch definiert werden - eine leere Zeile am Ende.

InsertWordBoundaries

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global dictionary
InitialiseDict dictionary dict.txt

}

proc AutoBuild {templ hier} {
global dictionary
 InsertWordBoundaries $templ $hier dictionary
 Word Canonic
}
```



Build Hierarchy



LevelFromDict

Prozeduraufruf mit Parametern

LevelFromDict $hier plevel clevel dictionary

plevel = die dominierende Ebene

clevel = die von plevel dominierte Ebene

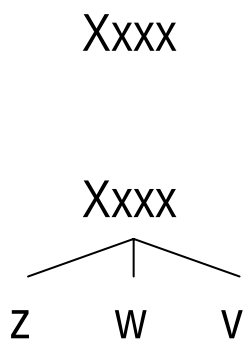
Benötigt als Vorbereitung die Prozedur **InitialiseDict**

Funktion

Erzeugt aus den Segmenten auf der Eltern-Ebene Segmente für eine Kind-Ebene.
Die neuen Segmente werden auf die Kind-Ebene geschrieben und mit dem Segment auf der Eltern-Ebene aus dem sie erzeugt wurden verbunden.
Kriterien für die Erzeugung sind in einem dictionary festgehaltene (selbsteditiert) Regeln.

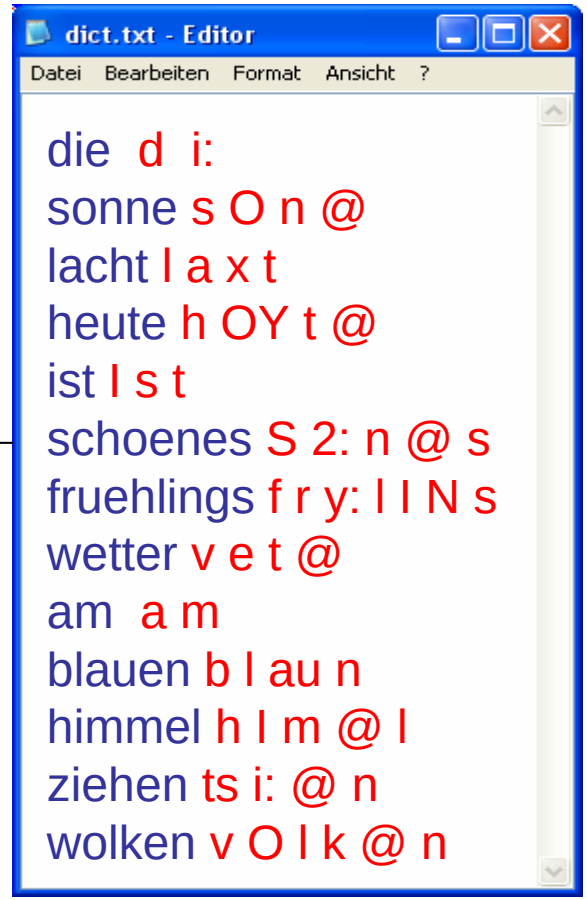
Dictionary

Xxxx z w v



LevelFromDict InitialiseDict

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global dictionary
InitialiseDict dictionary dict.txt

}

proc AutoBuild {templ hier} {
global dictionary
  LevelFromDict $hier Word Canonic dictionary
}
```

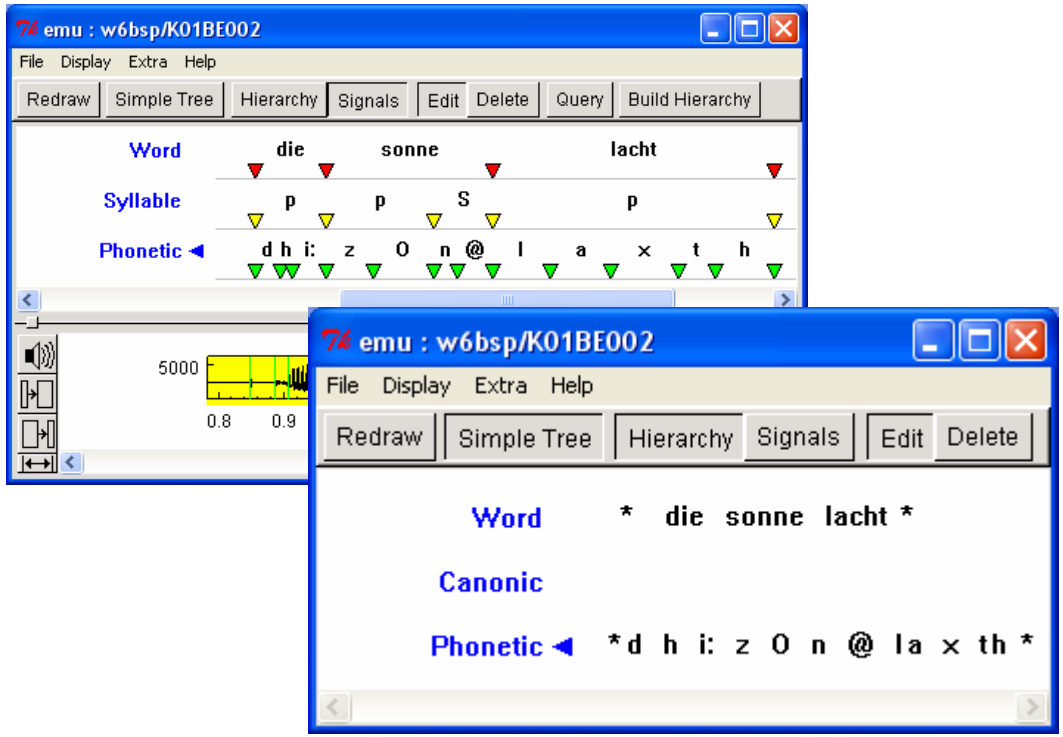


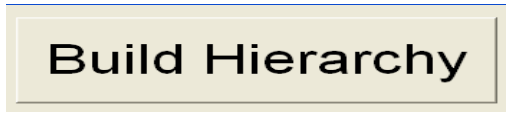
Die bereits vorhandenen Segmente auf der Elternebene

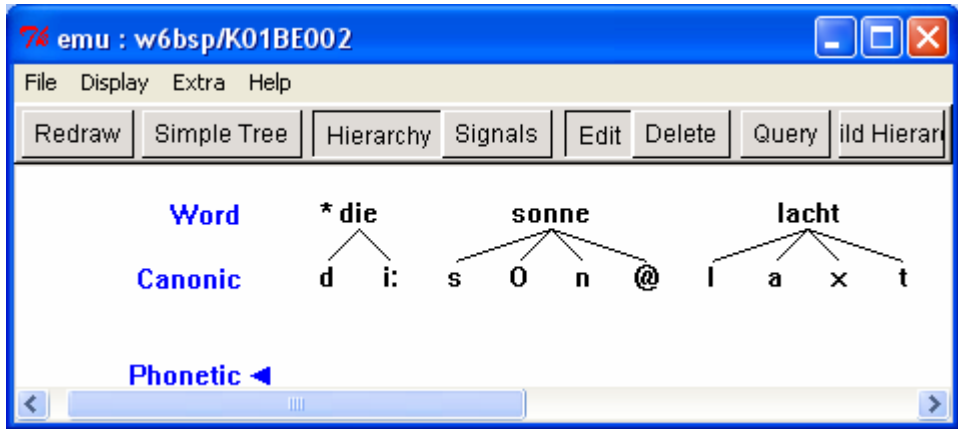
Segmente die für das Segment auf der Eltern-Ebene auf die Kind-Ebene geschrieben werden sollen

LevelFromDict

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global dictionary
InitialiseDict dictionary dict.txt

}

proc AutoBuild {templ hier} {
global dictionary
  LevelFromDict $hier Word Canonic dictionary
}
```







MapLevels

Prozeduraufruf mit Parametern

MapLevels $templ $hier plevel clevel rules

plevel = die dominierende Ebene

clevel = die von plevel dominierte Ebene

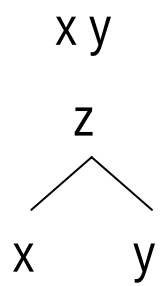
Benötigt als Vorbereitung die Prozedur **ReadLevelRules**

Funktion

Erzeugt aus den Segmenten auf der Kind-Ebene Segmente für eine Eltern-Ebene.
Die neuen Segmente werden auf die Eltern-Ebene geschrieben und mit dem Segment auf der Kind-Ebene aus dem sie erzeugt wurden verbunden.
Kriterien für die Erzeugung sind in einem Regelwerk (rules) festgehaltene (selbsteditiert) Regeln.

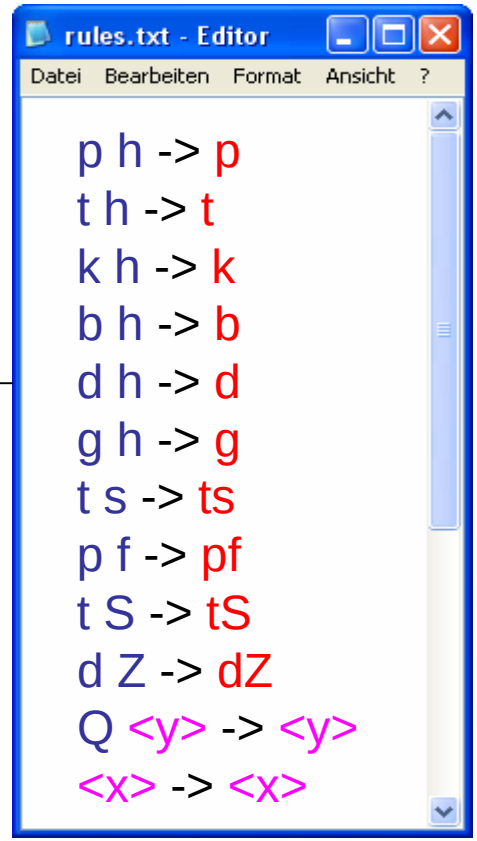
Dictionary

x y -> z



MapLevels ReadLevelRules

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global rules
set rules [ReadLevelRules rules.txt]


}

proc AutoBuild {templ hier} {
global rules
  MapLevels $templ $hier Phoneme Phonetic $rules

}
```



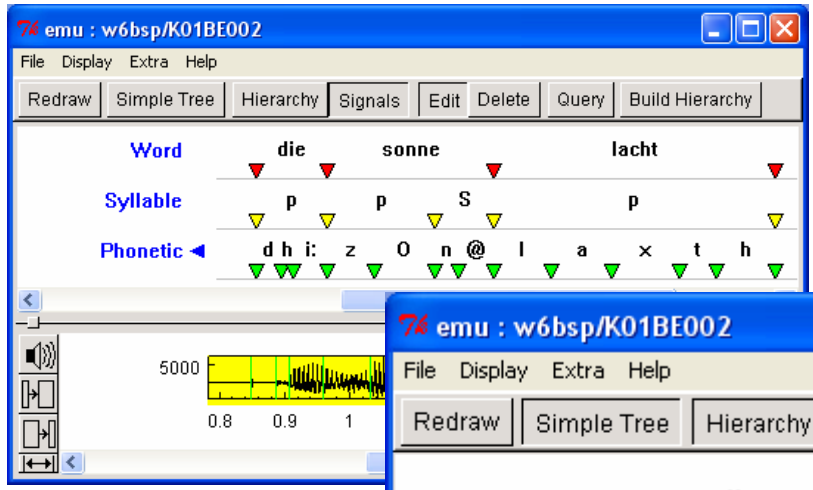
Die bereits vorhandenen Segmente auf der Kind-Ebene

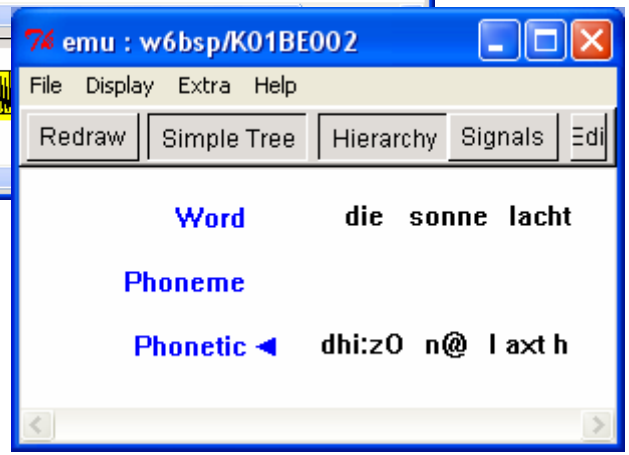
Segment das für das Segment auf der Kind-Ebene auf die Eltern-Ebene geschrieben werden sollen

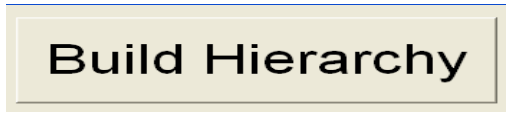
Wildcards – d.h. jedes Symbol (was keine andere Regel hat)

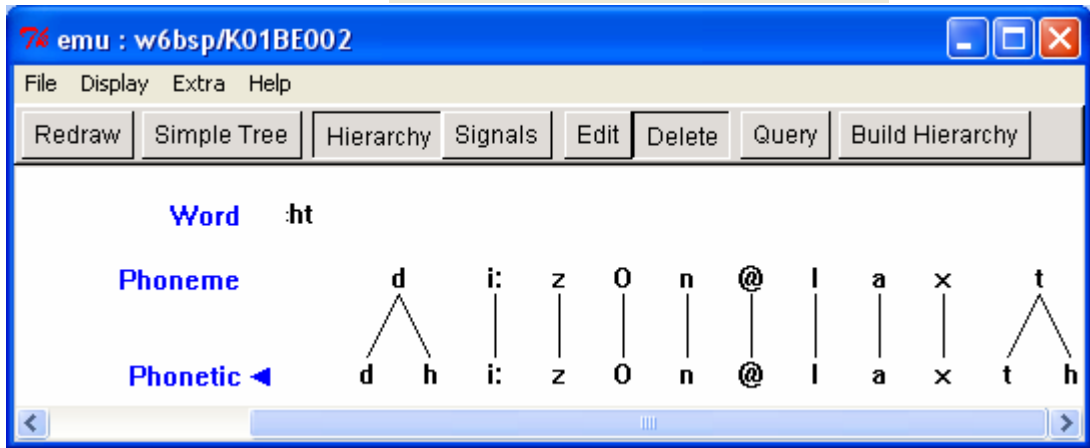
MapLevels

```
package require emu::autobuild

proc AutoBuildInit {templ} {
global rules
set rules [ReadLevelRules rules.txt]


}

proc AutoBuild {templ hier} {
global rules
  MapLevels $templ $hier Phoneme Phonetic $rules

}
```









Syllabify

Prozeduraufruf mit Parametern

Syllabify $templ $hier clevel plevel cons

plevel = die dominierende Ebene

clevel = die von plevel dominierte Ebene

Benötigt

cons - ein Array in proc AutoBuild

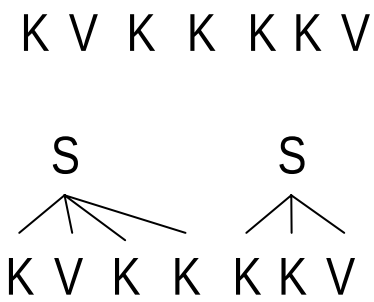
Legal Labels auf dem clevel in der Template-Datei

Funktion

Erzeugt aus den Segmenten auf der Kind-Ebene Segmente S für eine Eltern-Ebene.
Die neuen S Segmente werden auf die Eltern-Ebene geschrieben und mit dem Segment auf der Kind-Ebene aus dem sie erzeugt wurden verbunden.
Kriterien für die Erzeugung ist das “Maximal Onset” Prinzip. Dies richtet sich nach gültigen Segmenten, die als Nukleus dienen (Legal Labels) und legalen Silben-Onset Klustern (cons).

Durch cons erlaubt sind nur Onsetkluster von maximal 2 K.

Mögliche Nuklei sind V-legal labels der Klasse vowel.

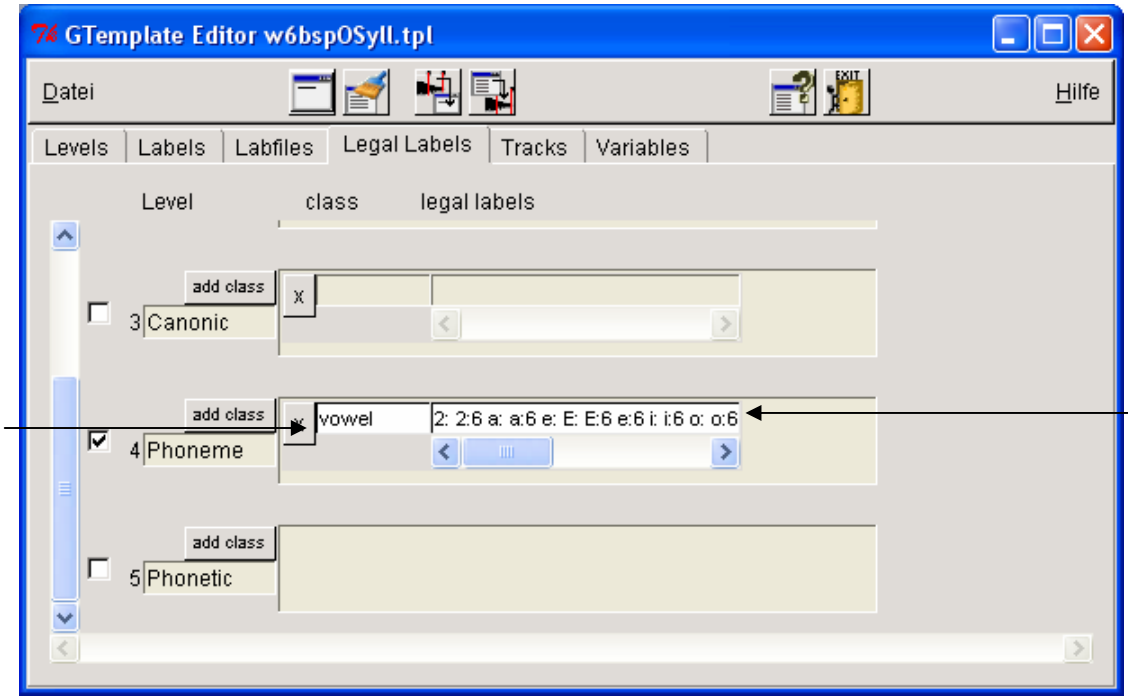


Syllabify Legal Labels

Legal Labels der Klasse *vowel* dienen als Nukleus

d.h. alle Phone(me), die in der Sprache als Nukleus vorkommen, müssen als Legal Label auf dem clevel in der Klasse *vowel* definiert werden

Klasse



Legal Labels einer Klasse – hier *vowel*

Für das Deutsche:

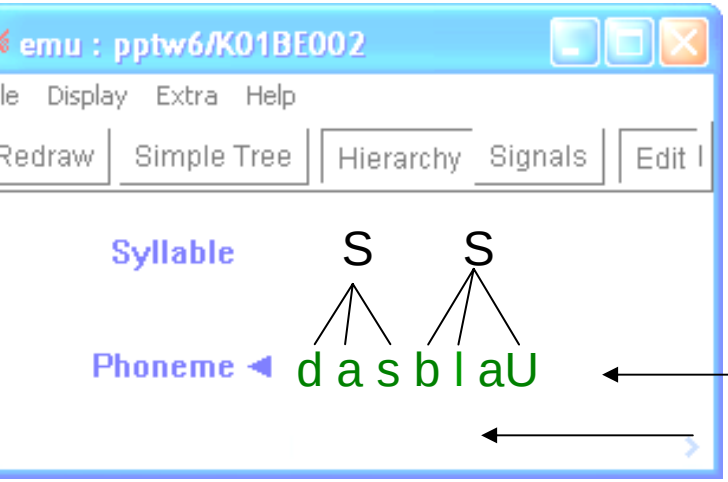
legal Phoneme vowel 2: 2:6 a: a:6 e: E: E:6 e:6 i: i:6 o: o:6 u: u:6 y: y:6 aI aU OY U 9 a a6 E E6 I I6 O O6 U6 Y Y6 6 @

Syllabify cons

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {

  set cons(triples) {pfl pfr tsv Spr Spl Str skr skl }

  set cons(pairs) {pr br tr dr kr gr fr Sr pl bl kl gl fl Sl Sm kn gn Sn kv Sv Sp St sk}

  set cons(singles) {p b t d k g pf ts dZ f v s z S Z C x m n N l r j h}

  Syllabify $templ $hier Phoneme Syllable cons

}
```

In cons werden alle in der Sprache zulässigen Onset-Kluster definiert.

```
package require emu::autobuild
proc AutoBuildInit {templ} { }
proc AutoBuild {templ hier} {
set cons(triples) {  }
set cons(pairs) { bl}
set cons(singles) {b l s d}
Syllabify $templ $hier Phoneme Syllable cons}
```



Fängt von hinten an

Sucht nach einem legal Phonem (einem Nukleus)

Ist nächster Konsonant ein legaler Onset?

Ist nächster Konsonant im Kluster ein legaler Onset?

Ist nächster Konsonant im Kluster ein legaler Onset?

Sucht nach einem legal Phonem (einem Nukleus)

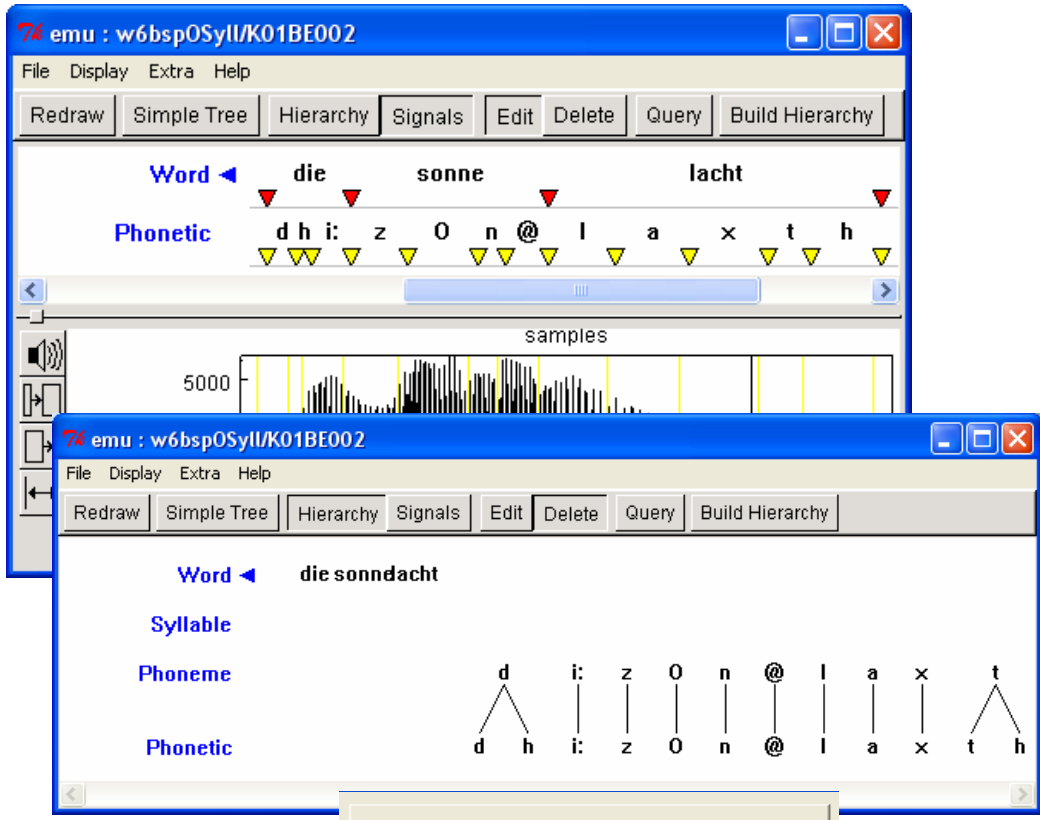
Ist nächster Konsonant ein legaler Onset?

Ist nächster Konsonant im Kluster ein legaler Onset?

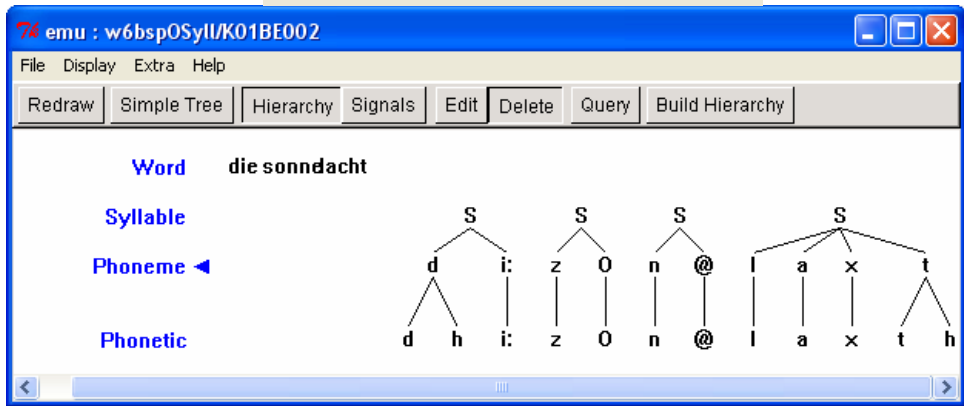
ENDE

Syllabify

```
package require emu::autobuild

proc AutoBuildInit {templ} {

}

proc AutoBuild {templ hier} {
  set cons(triples) {pfl pfr tsv Spr
Spl Str skr skl }
  set cons(pairs) {pr br tr dr kr gr
fr Sr pl bl kl gl fl Sl Sm kn gn Sn
kv Sv Sp St sk}
  set cons(singles) {p b t d k g pf
ts dZ f v s z S Z C x m n N l r j h}

  Syllabify $templ $hier Phoneme Syllable cons

}
```



Build Hierarchy



+ mögliche Nuklei in der Template-Datei als legal label der Klasse vowel definieren

Zusammenfassung

Linkt vorhandene Segmente miteinander

von Eltern-Ebene ausgehend zur Kind-Ebene

Linkt von der Zeit abhängig

LinkAllFromTimes
LinkFromTimes

Linkt von der Zeit unabhängig

InsertWordBoundaries
LinkSpans

Erzeugt neue Segmente aus einer vorhandenen Ebene

von Eltern-Ebene auf Kind-Ebene

LevelFromDict

von Kind-Ebene auf Eltern-Ebene

MapLevels

Syllabify