Forms of Prosody

Models – Maps – Defaults

2019-07-22, 14:30-16:30 Beijing, 08:30-10:30 Berlin

Dafydd Gibbon

Bielefeld University

Chinese Summer School:
Contemporary Phonetics and Phonology

Questions from Lecture 1:

Björn Lindblom, H&H theory

Hyper-hypo speech continuum

Hyperarticulation

- maximally distinct
- clear segments, syllables, prosody, ...
- formal speech
- slow speech
- hearer-oriented

Hypoarticulation

- scale of indistinctness
- reduced and deleted segments, syllable, prosody, ...
- informal speech
- fast speech
- speaker-oriented

Kul, Małgorzata. 2018. *Quantification and modelling of selected consonantal processes of casual speech in American English*. Habilitation thesis, Poznań: Adam Mickiewicz University.

Lindblom, Björn. 1990. Explaining Phonetic Variation: A Sketch of the H&H Theory. In Hardcastle, William J. and Alain Marchal, eds. *Speech Production and Speech Modelling*, Dordrecht: Kluwer, 403–439.

Is intonation universal?

Yes:

- pitch ranges, unmarked declination (downdrift), rising pitch non-terminal, falling pitch terminal
- rhythms at different frequencies
- prosodic syllables, words, phrases, ...

No:

- different functions
- changes according to
  - grammatical typology (especially morphology but also syntax)
  - lexical typology (phonemic and morphemic tones, pitch accent, stress)
  - pressure of cultural conventions (family, friends, school, media)

And: Is prosody learned or innate?

- babies hear prosody, heart, etc. before birth – tissues are a low frequency filter
- innateness arguments refer to grammar and ignore prosody and other factors.
- Poverty of the theory, not of the stimulus

Lecture 2: Method

Lecture 1: Qualitative, hermeneutic analysis, with reference to the semiotics of discoursal and musical patterns, on the basis of the Metalocutionary Theory of prosodic meaning.

Lecture 2: Qualitative, formal analysis, with discussion of the complexity of prosodic patterns, for example recursion, on the basis of different computational and other models.

Lecture 3: From qualitative to quantitative analysis of the sounds of rhythm and melody based on Rhythm Formant Theory, and using automatic analysis of speech signals from different discourse types and automatic classification of spoken discourse types.

In general, the procedure is exploratory and cross-disciplinary and oriented towards outlines and overviews, rather than narrowly confirmatory within a specific paradigm.

An exception is the last lecture!

Topics

Triadic semiotic theory, <meaning, form, sound>:

Meaning (Lecture 1: music, discourse, lexicon):

- Metalocutionary Theory: prosody points at times and locations in locutions

Form (Lecture 2):

- A computational phonological approach
- Linear Grammars, Templates: prosodic constructions*
- Temporal and spatial complexity of prosodic forms
- The recursion controversy
- Prosodic inheritance

Sound (Lecture 3):

- Rhythm Formant Theory: temporal structuring of speech at all ranks by
  - both sonority patterns and
  - fundamental frequency pattern
- Rhythm Formant Analysis software enables classification of language varieties according to speech rhythm – questions:
  - Are Rhythm Formants determinants of languages or speech styles and speech genres?
  - Properties of rhythm formants: frequency, bandwidth

Gras, Pedro and Wendy Elvira-Garcia. 2021. The role of intonation in Construction Grammar: on prosodic constructions. *Journal of Pragmatics*, 180, 232-247.

“There are many ways to do it” – Some Phonology Paradigms

There are many paradigms in prosody description: the European ‘tonetic’ school in applied linguistics, the US ‘phonemic tone levels’ school of Pike or Trager & Smith, and more recent generative, autosegmental, metrical and optimality theoretic approaches.

For example,

- ☞Prosodic Phonologies (Firth, etc., origins in Africanist linguistics)
- ☞Functionalist Prosodies (Halliday etc., origins in traditional grammar)
- ☞Generative Phonologies (Halle etc., origins in formal language theory and historical linguistics)
- ☞Autosegmental Phonologies (Goldsmith etc., origins in Africanist linguistics)
- ☞Metrical Phonologies (Liberman etc., origins in poetry)
- ☞Inheritance Network Phonologies (Gazdar etc., origins in default logic)
- ☞Optimality Phonologies (Smolensky etc., origins in biology)
- ☞Finite State Phonologies (Kay etc., origins in formal language theory and theoretical computer science)
- ☞Speech synthesis and recognition (Jelinek etc., origins in audio engineering)

And other traditions, for example,

- ☞the Chinese tradition of describing, for example, *syllables*, *tones*, *poetic patterning*
- ☞the Indian tradition of describing, for example, *sandhi*

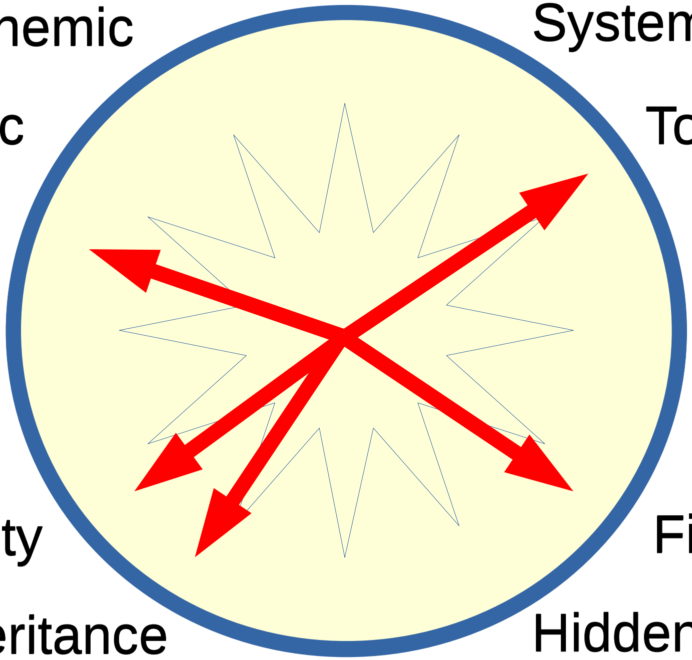
It is worth looking beyond ‘mainstream’ paradigms and models at other sources of inspiration. This is what I will be doing.

Orientation

Paradigms
A paradigm is a set of theories, models, methods, concepts and assumptions shared by a group of cooperating scientists.

Cf. Kuhn 1962

This lecture will mention a selection of the available theories and models:



Firthian

Phonemic

Systemic

Morphemic

Tonetic

Generative

Mereological

Autosegmental

Categorial

Metrical

Montague

Optimality

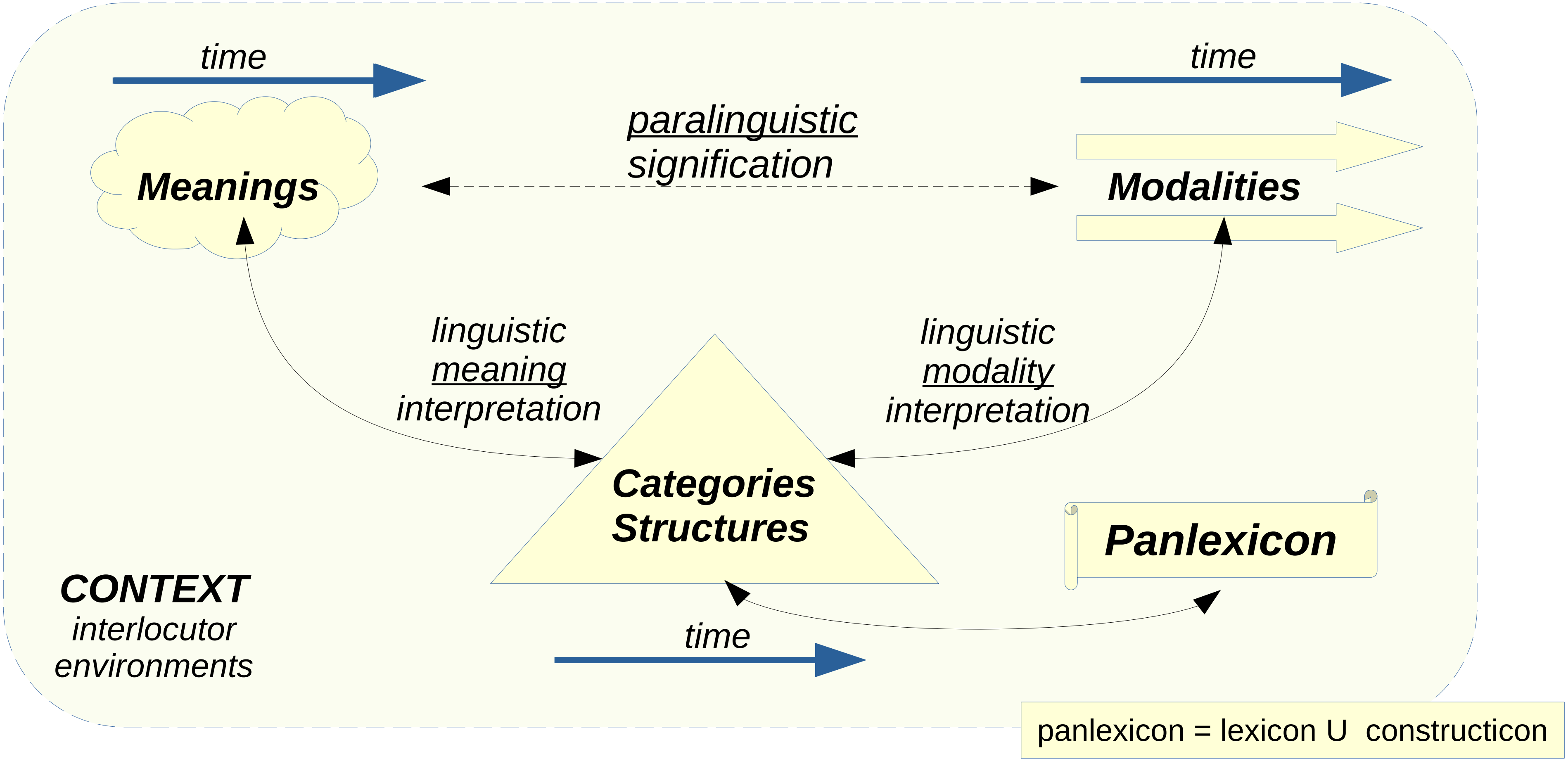
Finite State

Inheritance

Hidden Markov Models

Deep Neural Nets

Theoretical context: Semiotic Theory of Prosody



time

time

paralinguistic signification

Meanings

Modalities

linguistic meaning interpretation

linguistic modality interpretation

Categories Structures

Panlexicon

CONTEXT
interlocutor environments

time

panlexicon = lexicon U constructicon

Summary:

sign = *semiosis*(*time, structure*, *meaning*, *modality, context)*

structure = *order*(*time*, *phon*, *morph*, *syn, text*, *disc*)

meaning = *interpretation*(*time*, *structure*, *panlexicon*, *context*)

modality = *interpretation*(*time*, *voice, gesture)*

panlexicon = lexicon U constructicon

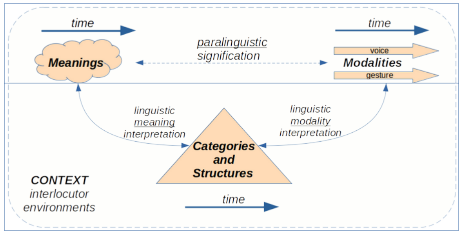
Rank-Interpretation Model of the Architecture of Speech

Ranks

Prosodic and Locutionary Signs

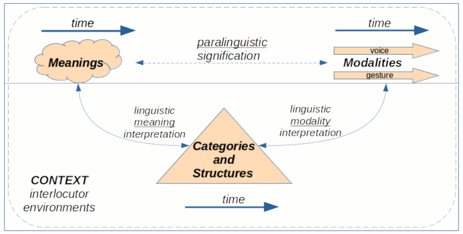
Prosodic Meanings as denotations

Dialogue



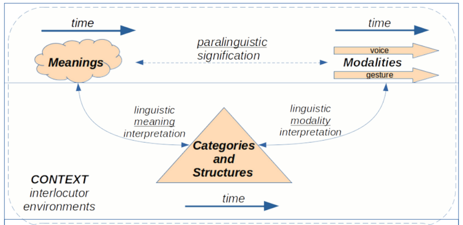
turn initiation (calling)
uptake securing
turn-taking, dialogue genres

Utterance



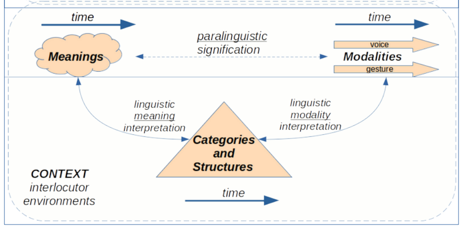
speech acts, (non)-finality
frequency-size code (Ohala)
cohesion: configuration, culmination, delimitatio
coordination with facial and hand gestures

Sentence



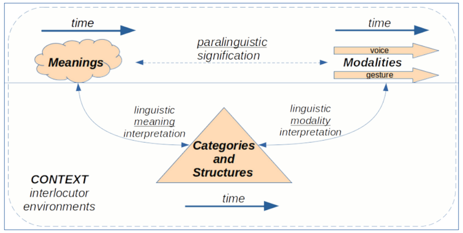
cohesion: configuration, culmination, delimitation
information structure: focus; theme-rheme; given-new
phrasal contrast, phrasal emphasis
subordination, parenthesis

Word
Morpheme



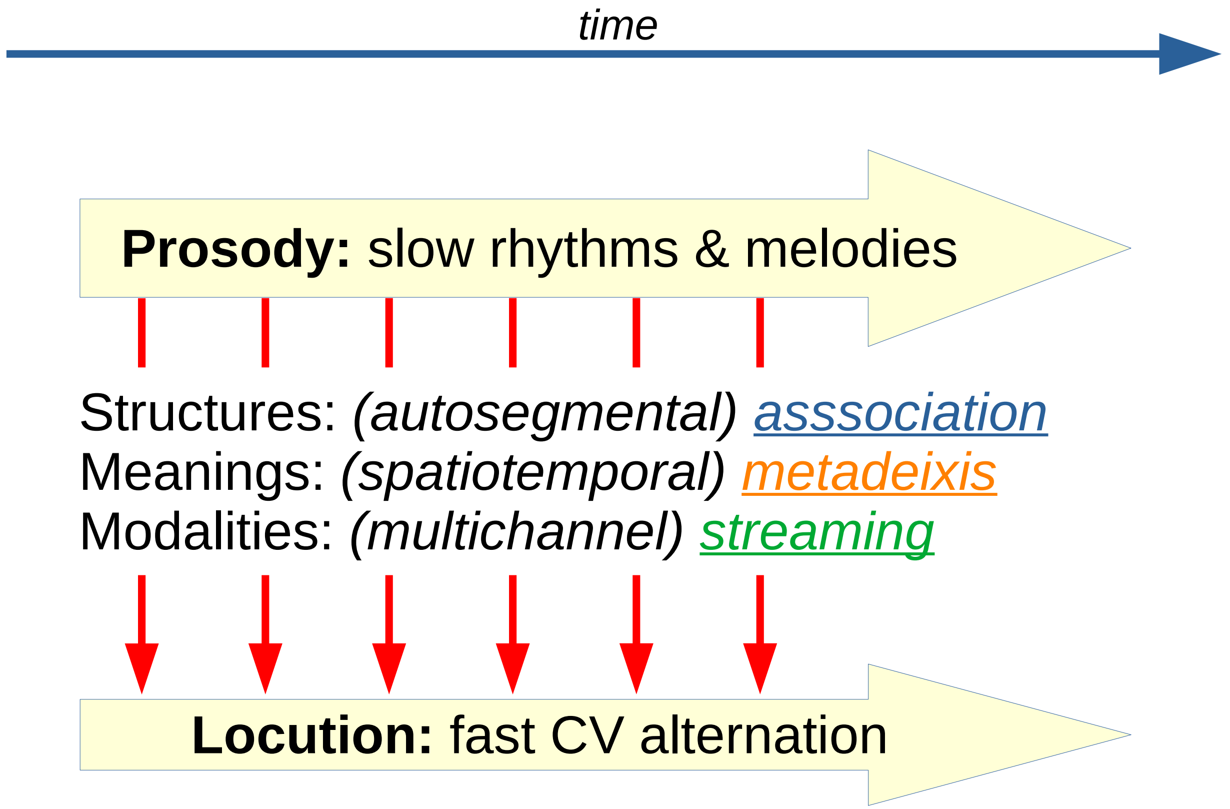
head-modifier relations in compound words
lexical contrast
lexical emphasis

Syllable
Phoneme



contrast with tones, pitch accents

Theoretical context: Metalocutionary theory


time
Prosody: slow rhythms & melodies
Structures: *(autosegmental)* asssociation
Meanings: *(spatiotemporal)* metadeixis
Modalities: *(multichannel)* streaming
Locution: fast CV alternation

Time Types:

cloud time (intuitive pre-theoretical everyday ‘real’ time)

clock time (Newtonian time, universal quantitative time: phonetics)

rubber time (Aristotelian time: Event &Articulatory Phonology, tree structures)

categorial time (abstract time points: phonology; duration contrast, context)

The syntax (= structure) of prosody

Basics, for prosody, too:

1. The forms of a language (morphemes, words, sentences, ...) are described by a grammar.
2. The components of a grammar:

   Vocabulary (Lexicon, Dictionary, Inventory)
   1. List of items (phonemes, morphemes, words, idioms, …)
   2. Set of paradigmatic (classificatory, similarity) relations

   Constructor (Rule system, Constraint system)
   1. Generator / Parser (creation and analysis of structures)
   2. Set of syntagmatic (compositional) relations
3. Compositional operations in prosody:
   1. Sequencing: concatenation of tokens (cf. standard phonologies & grammars)
   2. Parallelism: synchronisation; overlap (cf. autosegmental phonology)
   3. Grouping: generalisation; domain (cf. metrical phonology)

These operations are interpreted in terms of temporal relations

Theoretical context: event logics and interval algebras

Event logic relations such as the following (symbols modified):

Precedence: $A \prec B$

Immediate Precedence: A ^ B

Overlap: $A \circ B$

Include: $A \sqsubseteq B$

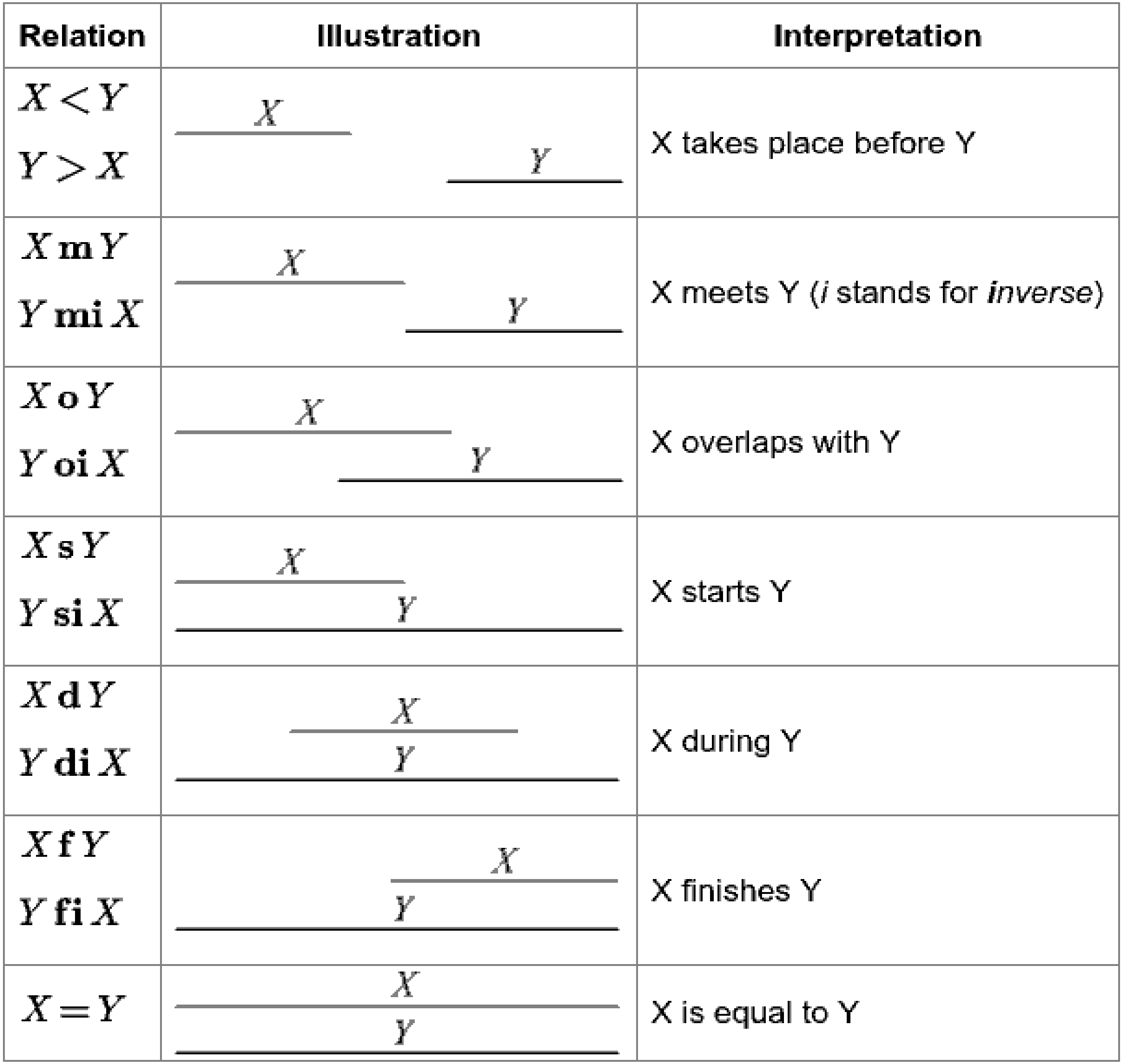
Ontological decision (cf. tiers in Praat):

1. points?
2. intervals?

Event Phonology (Steven Bird; Julie Carson-Berndsen)

Think of the interval tiers and point tiers in Praat TextGrids.

Theoretical context: Allen’s Interval Algebra



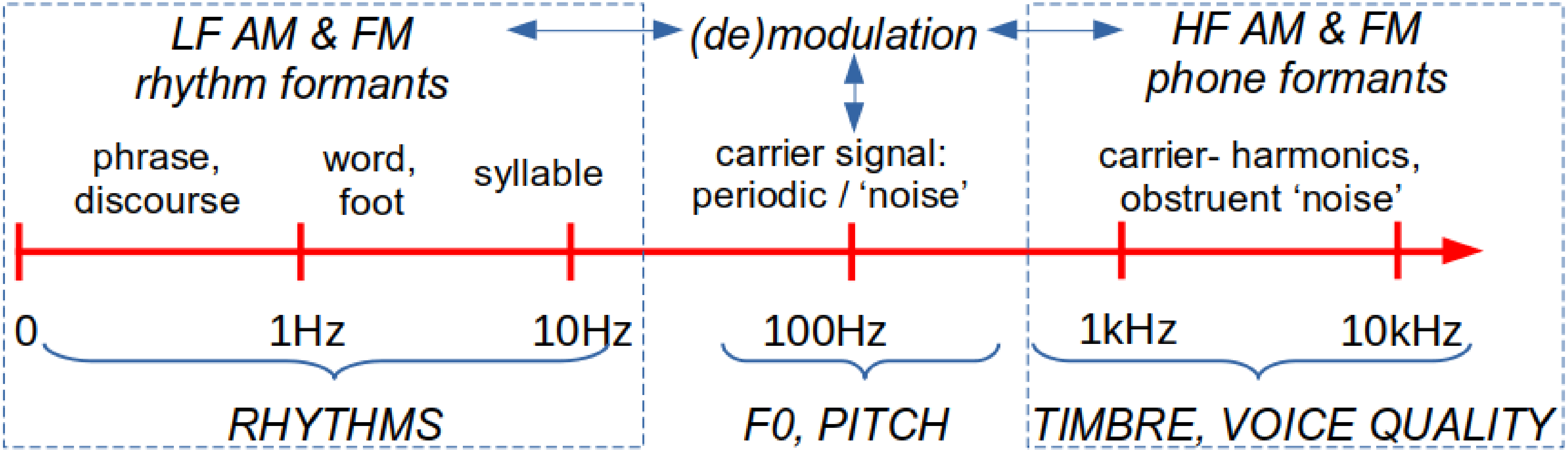
Relation	Illustration	Interpretation
$X < Y$ $Y > X$	X Y	X takes place before Y
$X\,\mathbf{m}\,Y$ $Y\,\mathbf{mi}\,X$	X Y	X meets Y (*i* stands for *inverse*)
$X\,\mathbf{o}\,Y$ $Y\,\mathbf{oi}\,X$	X Y	X overlaps with Y
$X\,\mathbf{s}\,Y$ $Y\,\mathbf{si}\,X$	X Y	X starts Y
$X\,\mathbf{d}\,Y$ $Y\,\mathbf{di}\,X$	X Y	X during Y
$X\,\mathbf{f}\,Y$ $Y\,\mathbf{fi}\,X$	X Y	X finishes Y
$X = Y$	X Y	X is equal to Y

Theoretical context: Modulation Code Theory

Low frequencies:
rhythm

Mid frequencies:
rhythm

High frequencies:
consonants and vowels



Low Frequency
AM and FM modulations

High Frequency
AM and FM modulations

A popular method for mapping linguistic units to phonetics

Annotation, a qualitative deductive-inductive method:

- segmentation and classification (‘labelling’) of prosodic forms such as:
  - consonantal and non-consonantal segment
  - syllable
  - foot
- the search for rhythm as isochrony* of similar units in sequence

*__isochrony__: equal clock timing, for example as an idealised phonetic interpretation of prosodic forms like syllables or stress groups

Problem: isochrony of similar units in sequence only a necessary condition on rhythm, not a sufficient condition.
Alternation of isochronous similar units in sequence is another necessary condition.

Both the Isochrony condition and the Alternation Condition together constitute a sufficient condition. Together they explain why rhythms have frequencies.

So the annotation method only describes ‘half’ of rhythm and does not explain it.
It is still a useful and popular method, but we need a more powerful method.

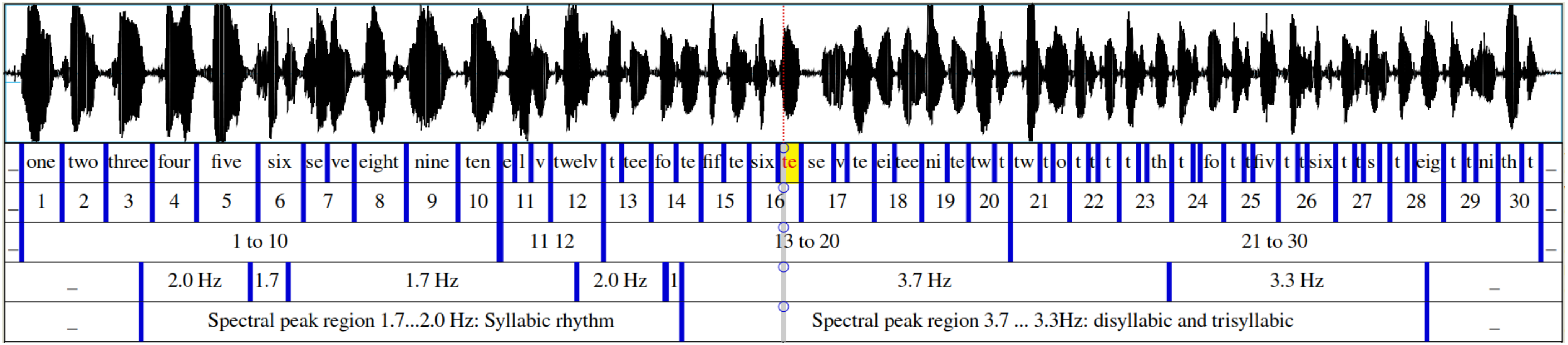
Annotation

Mapping forms to sounds – a qualitative approach

The qualitative annotation-based approach: procedure

1. Decide on a set of prosodically relevant forms:
   - phonetic, phonological
   - morphological, syntactic (part of speech, PoS tags)
   - semantic:
     - operator scope
     - information structure
   - pragmatic
     1. speech acts
     2. turn-taking
     3. discourse grammar
2. Annotation of relevant speech data
   - Search for and record data
   - Listen, transcribe, annotate
3. Calculate statistical properties
   - standard deviation, coefficient of variation, nPVI, ...

Event annotation with ‘Praat’: intervals and labels


_ one two three four five six se ve eight nine ten e l v twelv t tee fo te fif te six te se v te ei tee ni te tw t tw t o t t t t th t fo t t fiv t t six t t s t eig t t ni th t _
_ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 _
_ 1 to 10 11 12 13 to 20 21 to 30 _
_ 2.0 Hz 1.7 1.7 Hz 2.0 Hz 1 3.7 Hz 3.3 Hz _
_ Spectral peak region 1.7...2.0 Hz: Syllabic rhythm Spectral peak region 3.7 ... 3.3Hz: disyllabic and trisyllabic _

Download Praat

https://www.fon.hum.uva.nl/praat/
https://www.praat.org

Data

Pre-recording

Design systematic filenames
Design data scenario
Prepare equipment and participants
You can record with Praat or Audacity

Recording

record with proper distance (1 span)
enough to drink

Post-recording

save with systematic filename
archive systematically

Annotate with Praat

Read into Praat
Select “Annotation”
Annotate with prosodically relevant linguistic forms
Save Praat TextGrid format with systematic filename
Convert the Praat format to CSV spreadsheet format
This can be done easily with a Python script.

Analyse the spreadsheet file

With a spreadsheet.
With Python, R, MatLab, Stata, ...

Or analyse the Praat TextGrid file directly with TGA

Time Group Analyser online tool
http://wwwhomes.uni-bielefeld.de/gibbon/TGA/

Event annotation with ‘Praat’: intervals and labels

What you get is this, the TextGrid format:

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 11.017875
tiers? <exists>
size = 3
item []:
item [1]:
class = "IntervalTier"
name = "Syllables"
xmin = 0
xmax = 11.017875
intervals: size = 62
intervals [1]:
xmin = 0
xmax = 0.48339725121628835
text = "_"
intervals [2]:
xmin = 0.48339725121628835
xmax = 0.6964283269433246
text = "one"
intervals [3]:
xmin = 0.6964283269433246
xmax = 0.9009381596412812
text = "two"
intervals [4]:
xmin = 0.9009381596412812
xmax = 1.155155243342209
text = "three"
intervals [5]:
xmin = 1.155155243342209
xmax = 1.4091692796134065
text = "four"
intervals [6]:
xmin = 1.4091692796134065
xmax = 1.6293013911980108
text = "five"
```

Event annotation with ‘Praat’: intervals and labels

What you get is this, the TextGrid format:

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 11.017875
tiers? <exists>
size = 3
item []:
item [1]:
class = "IntervalTier"
name = "Syllables"
xmin = 0
xmax = 11.017875
intervals: size = 62
intervals [1]:
xmin = 0
xmax = 0.48339725121628835
text = "_"
intervals [2]:
xmin = 0.48339725121628835
xmax = 0.6964283269433246
text = "one"
intervals [3]:
xmin = 0.6964283269433246
xmax = 0.9009381596412812
text = "two"
intervals [4]:
xmin = 0.9009381596412812
xmax = 1.155155243342209
text = "three"
intervals [5]:
xmin = 1.155155243342209
xmax = 1.4091692796134065
text = "four"
intervals [6]:
xmin = 1.4091692796134065
xmax = 1.6293013911980108
text = "five"
```

What you need is this, the CSV format:

File	Tier	Label	Start	End	Duration
one-to-thirty-11s_16k	Syllables	_	0.000	0.249	0.249
one-to-thirty-11s_16k	Syllables	_	0.249	0.483	0.234
one-to-thirty-11s_16k	Syllables	one	0.483	0.696	0.213
one-to-thirty-11s_16k	Syllables	two	0.696	0.901	0.205
one-to-thirty-11s_16k	Syllables	three	0.901	1.155	0.254
one-to-thirty-11s_16k	Syllables	four	1.155	1.409	0.254
one-to-thirty-11s_16k	Syllables	five	1.409	1.629	0.220
one-to-thirty-11s_16k	Syllables	six	1.629	1.883	0.254
one-to-thirty-11s_16k	Syllables	se	1.883	2.020	0.137
one-to-thirty-11s_16k	Syllables	ven	2.020	2.148	0.128
one-to-thirty-11s_16k	Syllables	eight	2.148	2.328	0.180
one-to-thirty-11s_16k	Syllables	nine	2.328	2.551	0.223
one-to-thirty-11s_16k	Syllables	ten	2.551	2.751	0.200
one-to-thirty-11s_16k	Syllables	e	2.751	2.821	0.070
one-to-thirty-11s_16k	Syllables	le	2.821	2.936	0.115
one-to-thirty-11s_16k	Syllables	ven	2.936	3.020	0.084
one-to-thirty-11s_16k	Syllables	twelve	3.020	3.296	0.276
one-to-thirty-11s_16k	Syllables	thir	3.296	3.461	0.165
one-to-thirty-11s_16k	Syllables	teen	3.461	3.615	0.154
one-to-thirty-11s_16k	Syllables	four	3.615	3.764	0.149
one-to-thirty-11s_16k	Syllables	teen	3.764	3.921	0.157
one-to-thirty-11s_16k	Syllables	fif	3.921	4.056	0.135
one-to-thirty-11s_16k	Syllables	teen	4.056	4.222	0.166
one-to-thirty-11s_16k	Syllables	six	4.222	4.449	0.227
one-to-thirty-11s_16k	Syllables	teen	4.449	4.547	0.098
one-to-thirty-11s_16k	Syllables	se	4.547	4.680	0.133
one-to-thirty-11s_16k	Syllables	ven	4.680	4.748	0.068
one-to-thirty-11s_16k	Syllables	teen	4.748	4.920	0.172
one-to-thirty-11s_16k	Syllables	eigh	4.920	5.025	0.105
one-to-thirty-11s_16k	Syllables	teen	5.025	5.208	0.183
one-to-thirty-11s_16k	Syllables	nine	5.208	5.356	0.148
one-to-thirty-11s_16k	Syllables	teen	5.356	5.506	0.150
one-to-thirty-11s_16k	Syllables	twen	5.506	5.734	0.228
one-to-thirty-11s_16k	Syllables	ty	5.734	5.863	0.129
one-to-thirty-11s_16k	Syllables	twen	5.863	6.036	0.173
one-to-thirty-11s_16k	Syllables	ny	6.036	6.100	0.064
one-to-thirty-11s_16k	Syllables	one	6.100	6.230	0.130
one-to-thirty-11s_16k	Syllables	twen	6.230	6.432	0.202
one-to-thirty-11s_16k	Syllables	ty	6.432	6.550	0.118
one-to-thirty-11s_16k	Syllables	two	6.550	6.703	0.153
one-to-thirty-11s_16k	Syllables	twen	6.703	6.896	0.193
one-to-thirty-11s_16k	Syllables	ty	6.896	6.959	0.063
one-to-thirty-11s_16k	Syllables	three	6.959	7.132	0.173
one-to-thirty-11s_16k	Syllables	twen	7.132	7.321	0.189
one-to-thirty-11s_16k	Syllables	ty	7.321	7.407	0.086
one-to-thirty-11s_16k	Syllables	four	7.407	7.561	0.154
one-to-thirty-11s_16k	Syllables	twen	7.561	7.741	0.180
one-to-thirty-11s_16k	Syllables	ty	7.741	7.793	0.052
one-to-thirty-11s_16k	Syllables	five	7.793	8.003	0.210
one-to-thirty-11s_16k	Syllables	twen	8.003	8.192	0.189
one-to-thirty-11s_16k	Syllables	ty	8.192	8.239	0.047
one-to-thirty-11s_16k	Syllables	six	8.239	8.477	0.238
one-to-thirty-11s_16k	Syllables	twen	8.477	8.674	0.197
one-to-thirty-11s_16k	Syllables	sen	8.674	8.903	0.229
one-to-thirty-11s_16k	Syllables	twen	8.903	9.071	0.168
one-to-thirty-11s_16k	Syllables	ny	9.071	9.174	0.103
one-to-thirty-11s_16k	Syllables	eight	9.174	9.302	0.128
one-to-thirty-11s_16k	Syllables	twen	9.302	9.462	0.160
one-to-thirty-11s_16k	Syllables	ny	9.462	9.559	0.097
one-to-thirty-11s_16k	Syllables	nine	9.559	9.745	0.186
one-to-thirty-11s_16k	Syllables	thir	9.745	9.996	0.251
one-to-thirty-11s_16k	Syllables	ty	9.996	10.151	0.155
one-to-thirty-11s_16k	Syllables	_	10.151	11.018	0.867

textgridtier2csv.py

```
#!/usr/bin/python
# textgridtier2csv.py D. Gibbon 2015.02.12

# Convert a Praat TextGrid tier to CSV  format

#------------------------------------------------------------
# Import standard modules

import os, re, sys

#------------------------------------------------------------
# Input TextGrid from CLII

if len(sys.argv) < 3:
    print("Usage:",sys.argv[0],'<filename> <tiername>')
    exit()
fname = sys.argv[1]
tname = sys.argv[2]
if not os.path.isfile(fname):
    print("File",fname,"does not exist.")
    exit()

textgrid = open(fname,'r').read().split('\n')
fname = sys.argv[1].split('.')[0]

#------------------------------------------------------------
# Remove initial and final spaces

nugrid = []
for l in textgrid:
    a = ''
    l = re.sub(' *$','',l)
    l = re.sub('^ *','',l)
    l = re.sub('\"','',l)
    if l != '':
        nugrid += [l]
```

```
def extracttiers(nugrid,outflag):
    tierkey = ''
    returnstring = ''
    output = ''
    val = ''
    start = 0
    if not outflag in ['file','string']:
        tierkey = outflag
    for i in range(len(nugrid)):
        l1 = nugrid[i].split(' = ')
        if len(l1) > 1:
            val = l1[1]
        if val == 'IntervalTier':
                if start > 0:
                    if tierkey == tiername:
                        return output
                    if outflag == 'file':
                        open(fname+'-'+tiername+'.csv','w').write(output)
                    returnstring += output
                output = ''
                tiername = nugrid[i+1].split(' = ')[1]
                start = 1
        l2 = nugrid[i].split(' ')
        if l2[0] == 'intervals':
            xmin = "%.3f"%float(nugrid[i+1].split(' = ')[1])
            xmax = "%.3f"%float(nugrid[i+2].split(' = ')[1])
            text = nugrid[i+3].split(' = ')[1]
            dur = "%.3f"%(float(xmax)-float(xmin))
            interval = "\t".join([fname,tiername,text,xmin,xmax,dur])+"\n"

#           interval = fname+'\t'+tiername+'\t'+text+'\t'+"%.3f"%xmin+'\
t'+"%.3f"%xmax+'\t'+"%.3f"%dur+'\n'
            output += interval
    if outflag == 'file':
        open(fname+'-'+tiername+'.csv','w').write(output)
    if outflag == 'string':
        returnstring += output
        return returnstring
    else:
        return ''
print(extracttiers(nugrid,tname))
```

Interval analysis and PVI – the search for isochrony

$$rPVI(D) = \sum |d_k - d_{k+1}| / (n-1)$$

$$nPVI(D) = 100 \times \sum \left| \frac{d_k - d_{k+1}}{(d_k + d_{k+1})/2} \right| / (n-1)$$

raw Pairwise Variability Index

normalised Pairwise Variability Index

The measure defines an overall ‘next-door neighbour distance’.

A distance measure compares two ordered sequences (vectors).

So to understand the *nPVI* as a distance measure, the sequence of durations needs to be separated into two sequences.

This would be done by making a copy of the sequence, removing the first element of one sequence and the last element of the other, and using the two sequences for distance comparison.

Actually any distance measure could be used, for example Euclidean Distance, or Cosine Distance. A study of these measures with annotation data would make a nice B.A. or even M.A. thesis.

Interval analysis and PVI – the search for isochrony

$$rPVI(D)=\sum\left|d_k-d_{k+1}\right|/(n-1)$$

$$nPVI(D)=100\times\sum\left|\frac{d_k-d_{k+1}}{(d_k+d_{k+1})/2}\right|/(n-1)$$

raw Pairwise Variability Index

normalised Pairwise Variability Index

The measure defines an overall ‘next-door neighbour distance’:

Similarity to Manhattan Distance

Similarity to Canberra Distance (Normalised Manhattan Distance)

$$MD(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

$$NormMD(x,y)=\sum_{i=1}^{n}\frac{|x_i-y_i|}{|x_i|+|y_i|}$$

Interval analysis and PVI – the search for isochrony

$$rPVI(D) = \sum \left| d_k - d_{k+1} \right| / (n-1)$$

$$nPVI(D) = 100 \times \sum \left| \frac{d_k - d_{k+1}}{(d_k + d_{k+1})/2} \right| / (n-1)$$

raw Pairwise Variability Index

normalised Pairwise Variability Index

The measure defines an overall ‘next-door neighbour distance’.

durations:	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9

Interval analysis and PVI – the search for isochrony

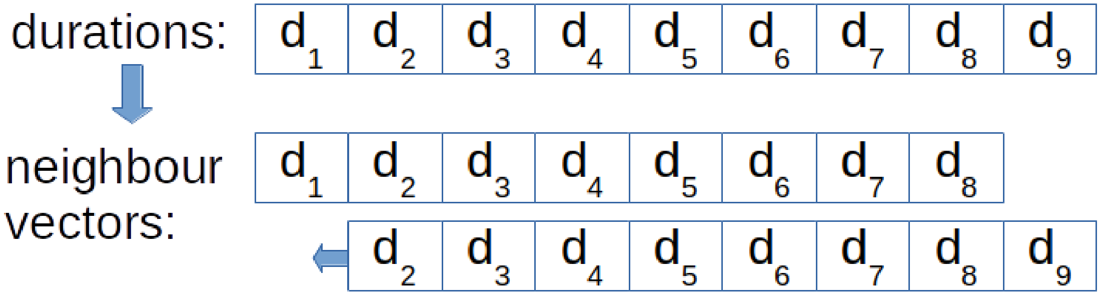
$$rPVI(D) = \sum |d_k - d_{k+1}| / (n-1)$$

$$nPVI(D) = 100 \times \sum \left| \frac{d_k - d_{k+1}}{(d_k + d_{k+1})/2} \right| / (n-1)$$

raw Pairwise Variability Index

normalised Pairwise Variability Index

The measure defines an overall ‘next-door neighbour distance’.



durations: d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9

neighbour vectors: d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8

d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9

To understand the *nPVI* as a distance measure (Canberra Distance):

1. Make a copy of the duration sequence from the annotation.
2. Remove the last duration from the first and the first from the second sequence.
3. Align the two sequences.
4. Calculate the average of all absolute differences (divided by their average) of the aligned duration pairs.
5. Multiply by 100 (this is sugar on the cake, not essential).

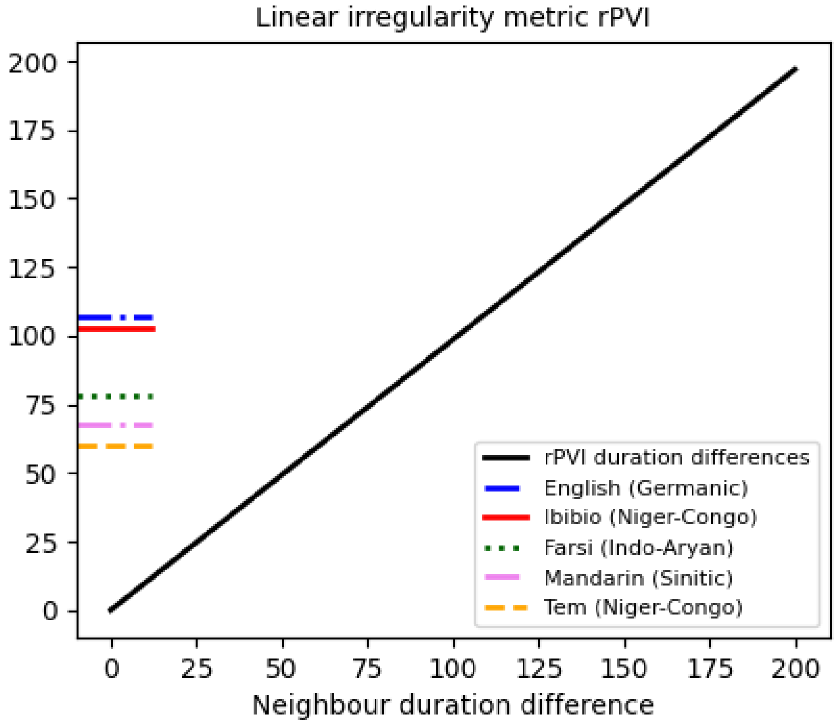
Interval analysis and PVI – the search for isochrony

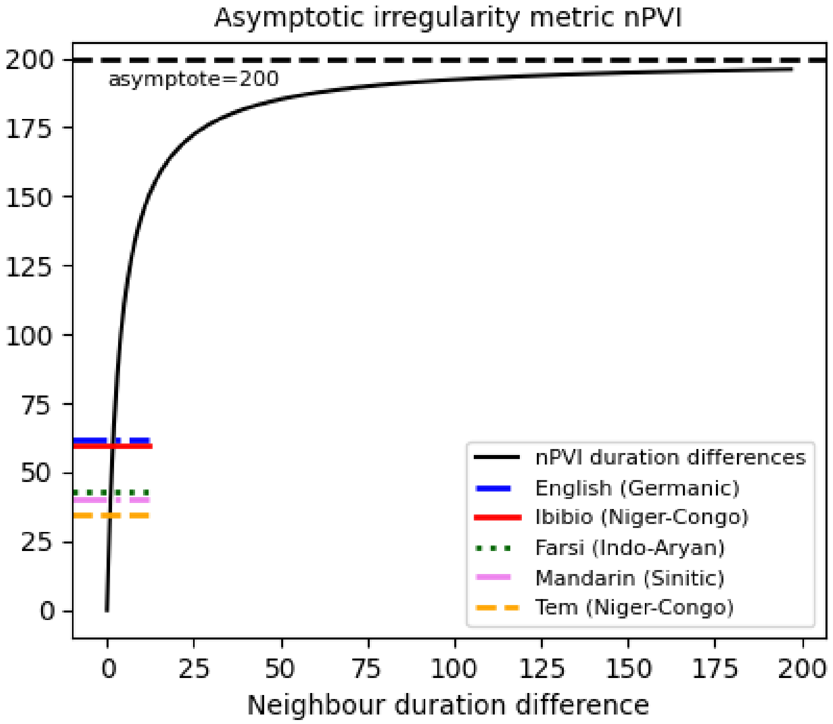
$$rPVI\left(D\right)=\sum\left|d_k-d_{k+1}\right|/\left(n-1\right)$$

$$nPVI\left(D\right)=100\times\sum\left|\frac{d_k-d_{k+1}}{\left(d_k+d_{k+1}\right)/2}\right|/\left(n-1\right)$$

raw Pairwise Variability Index

normalised Pairwise Variability Index





Assessment of interval duration measures

The interval duration measures can be useful heuristic measures.

They have the following properties:

1. the procedure is a hybrid qualitative (annotation) and quantitative (statistical analysis) procedure:
    - through the annotation procedure the signal is filtered through the perceptual skills of an annotator and the signal is not analysed directly
2. the procedure ignores the alternation property of rhythm by using absolute values, (which gives the same values for positive and negative differences between neighbours)
3. they are often called ‘rhythm metrics’, but this is an exaggeration:

    the interval duration measures calculate irregularity, not rhythmicality;

Conclusion:

The ‘irregularity measures’ do not provide a model, or a theory, or an explanation of rhythm.

A more powerful theory and method are necessary, in addition to the irregularity measures.

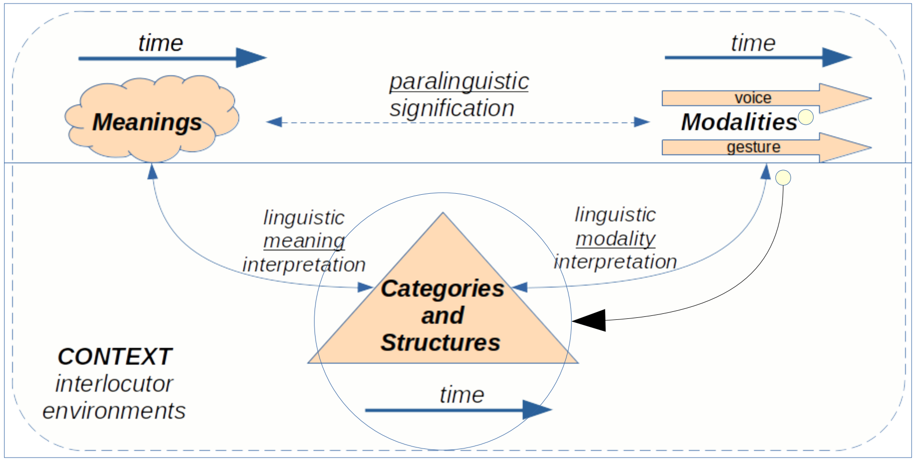
Forms and sounds: looking ahead to Sunday’s lecture

Rhythm Formant Theory
(RFT)

+

Rhythm Formant Analysis
(RFA)

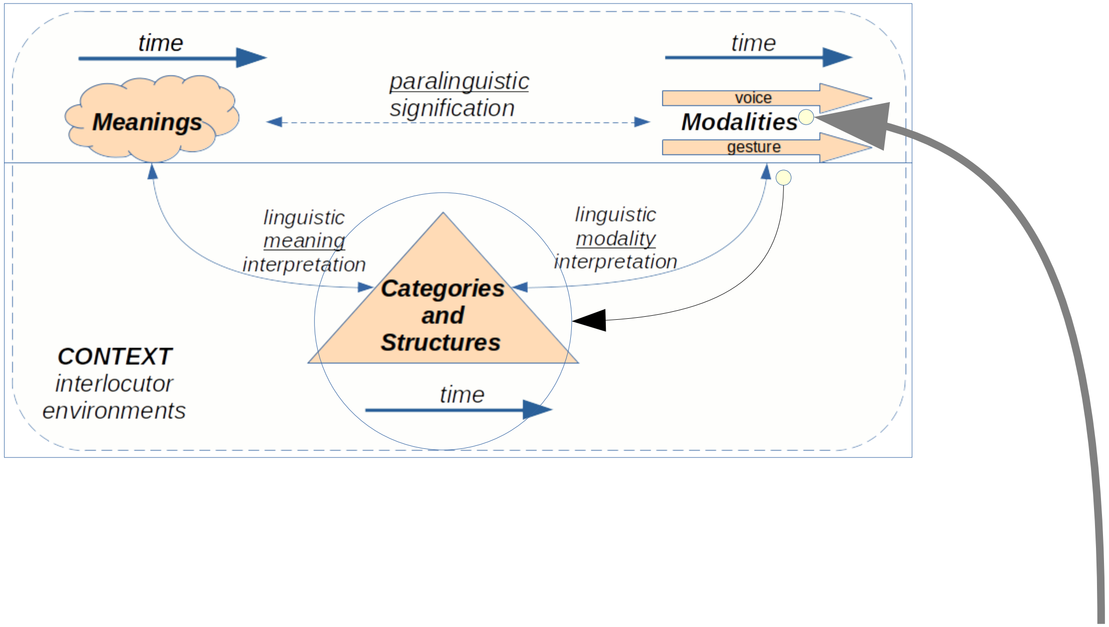
Rhythm Formants and their Structural Correlates



Rhythm Formant Analysis:

1. Low pass signal smoothing
2. Envelope extraction:
   1. AM: signal rectification
   2. FM: F0 estimation
3. Fourier analysis:
   1. AM LF spectrum & spectrogram
   2. FM LF spectrum & spectrogram
4. Cluster analysis

Rhythm Formants and their Structural Correlates



time

time

paralinguistic signification

Meanings

voice

Modalities

gesture

linguistic meaning interpretation

linguistic modality interpretation

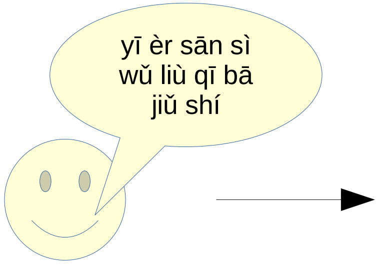
Categories and Structures

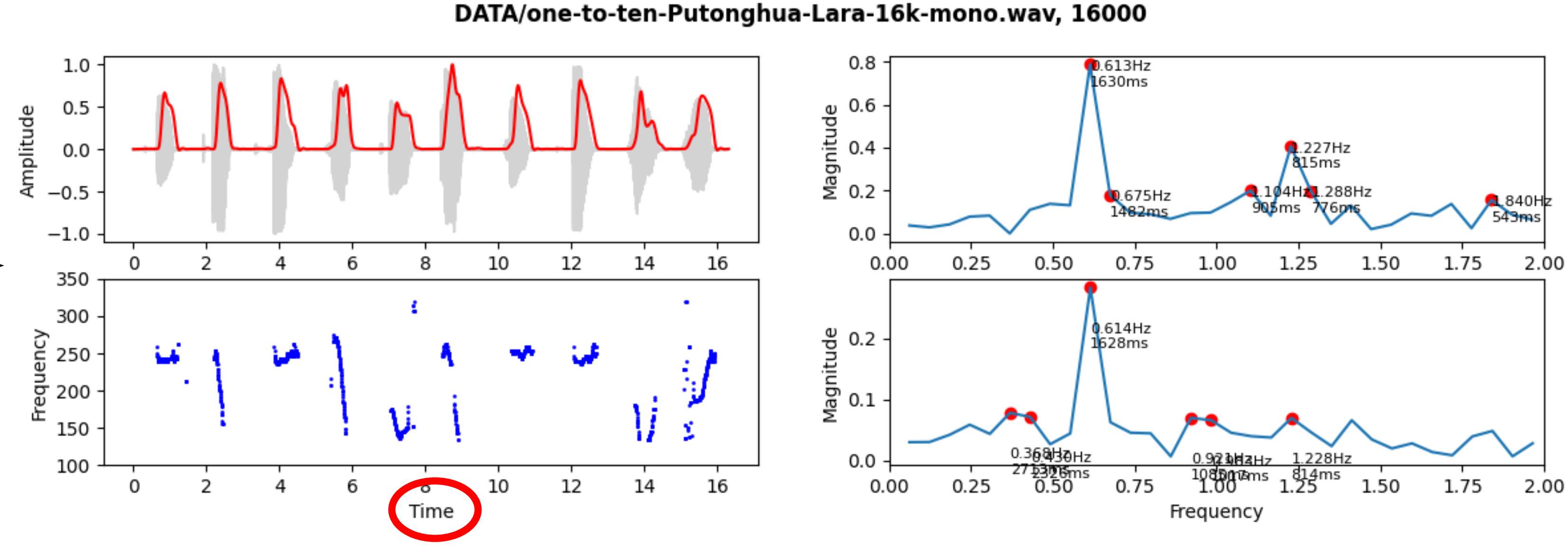
CONTEXT
interlocutor
environments

time

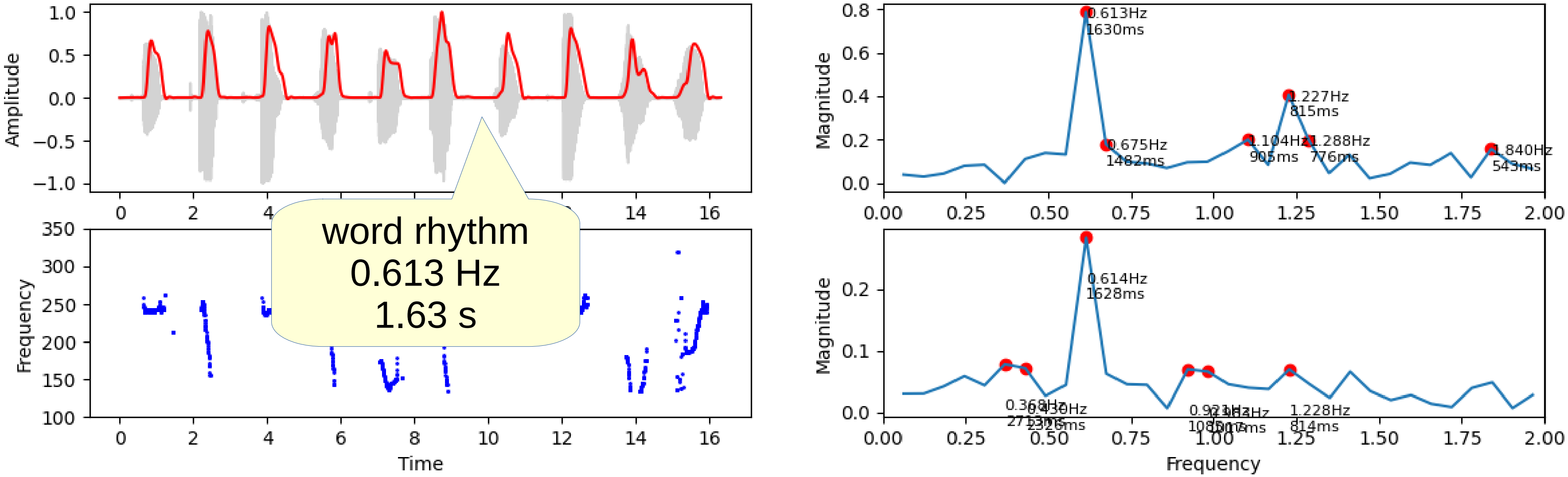
Rhythm Formant Analysis:

1. Low pass signal smoothing
2. Envelope extraction:
    1. AM: signal rectification
    2. FM: F0 estimation
3. Fourier analysis:
    1. AM LF spectrum & spectrogram
    2. FM LF spectrum & spectrogram
4. Cluster analysis



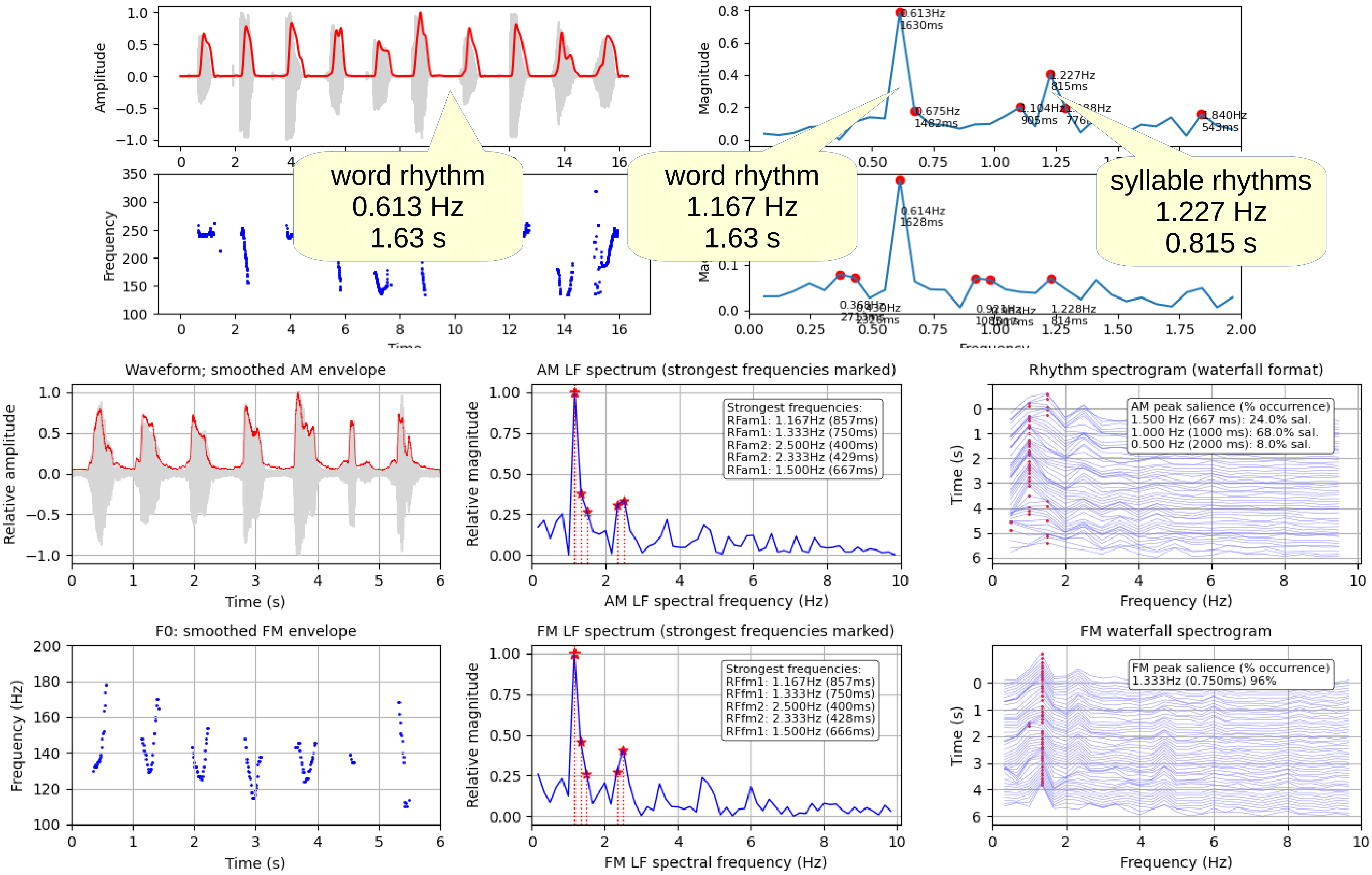


Rhythm Formants and their Structural Correlates



word rhythm
0.613 Hz
1.63 s

Rhythm Formants and their Structural Correlates



word rhythm
0.613 Hz
1.63 s

word rhythm
1.167 Hz
1.63 s

syllable rhythms
1.227 Hz
0.815 s

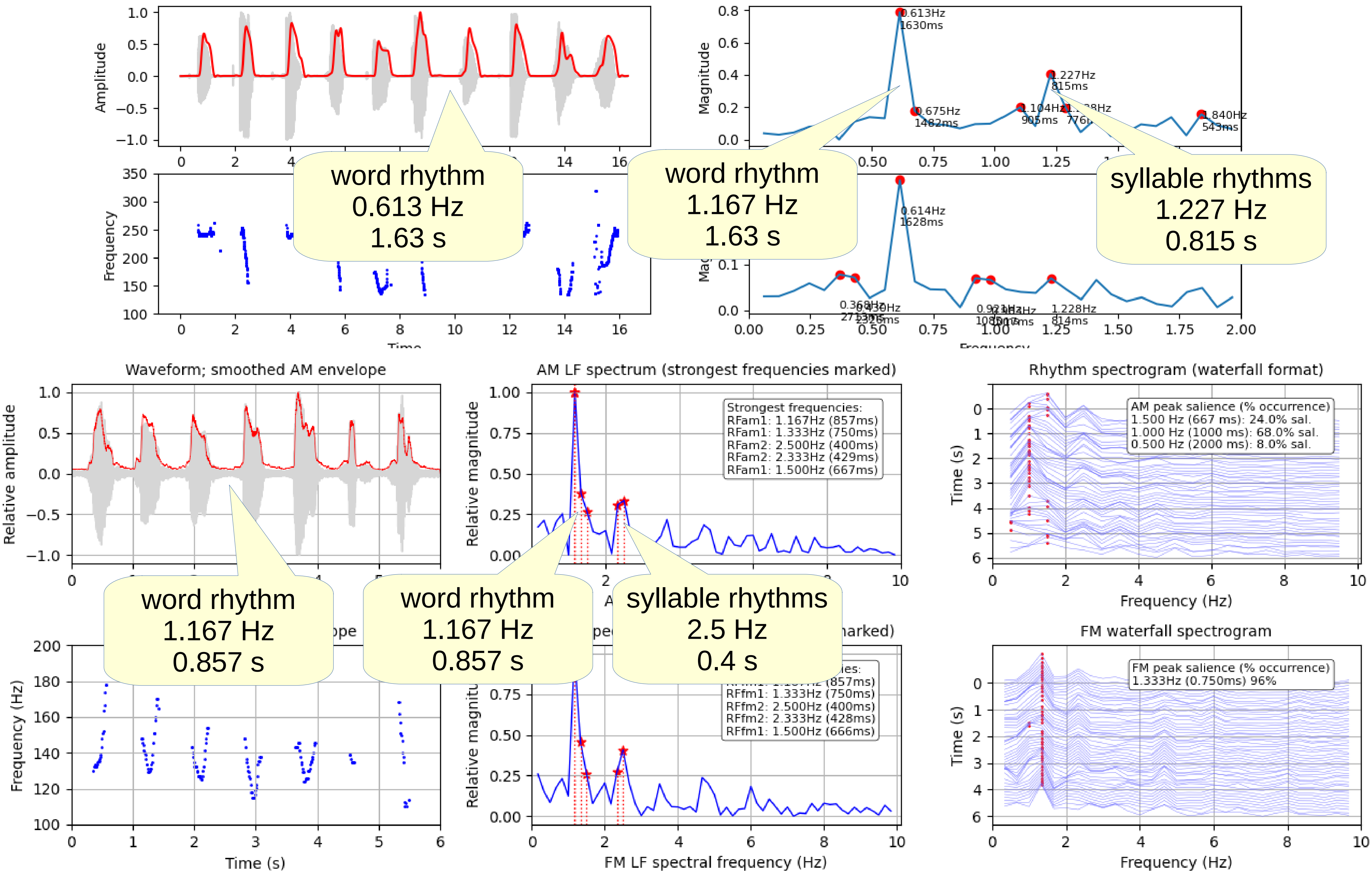
Strongest frequencies:
RFam1: 1.167Hz (857ms)
RFam1: 1.333Hz (750ms)
RFam2: 2.500Hz (400ms)
RFam2: 2.333Hz (429ms)
RFam1: 1.500Hz (667ms)

AM peak salience (% occurrence)
1.500 Hz (667 ms): 24.0% sal.
1.000 Hz (1000 ms): 68.0% sal.
0.500 Hz (2000 ms): 8.0% sal.

Strongest frequencies:
RFfm1: 1.167Hz (857ms)
RFfm1: 1.333Hz (750ms)
RFfm2: 2.500Hz (400ms)
RFfm2: 2.333Hz (428ms)
RFfm1: 1.500Hz (666ms)

FM peak salience (% occurrence)
1.333Hz (0.750ms) 96%

Rhythm Formants and their Structural Correlates



word rhythm
0.613 Hz
1.63 s

word rhythm
1.167 Hz
1.63 s

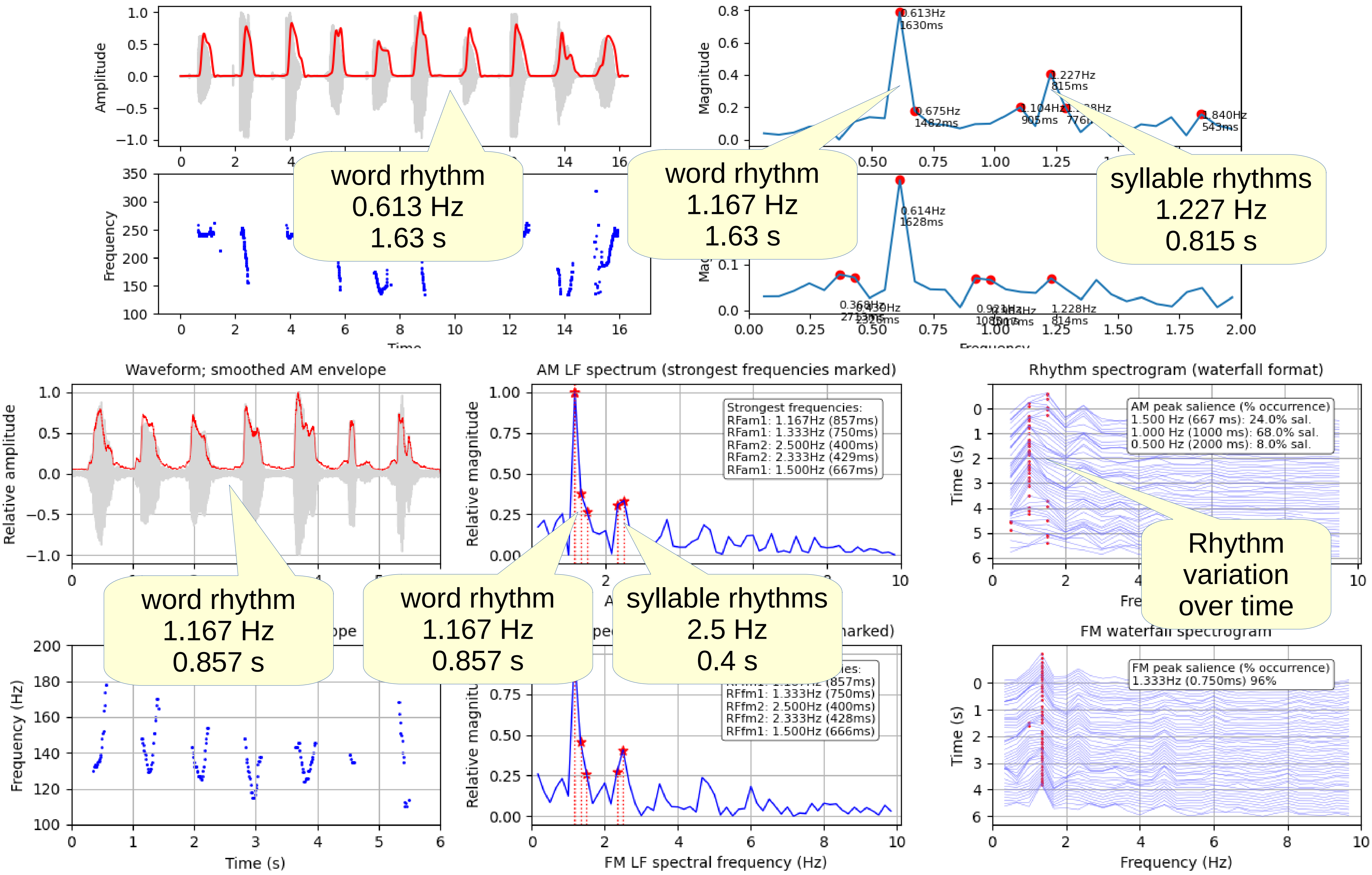
syllable rhythms
1.227 Hz
0.815 s

word rhythm
1.167 Hz
0.857 s

word rhythm
1.167 Hz
0.857 s

syllable rhythms
2.5 Hz
0.4 s

Rhythm Formants and their Structural Correlates



word rhythm
0.613 Hz
1.63 s

word rhythm
1.167 Hz
1.63 s

syllable rhythms
1.227 Hz
0.815 s

Rhythm
variation
over time

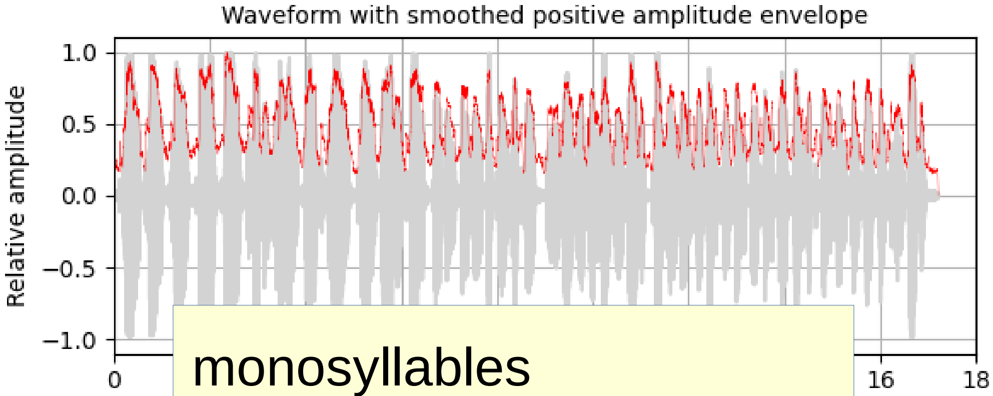
word rhythm
1.167 Hz
0.857 s

word rhythm
1.167 Hz
0.857 s

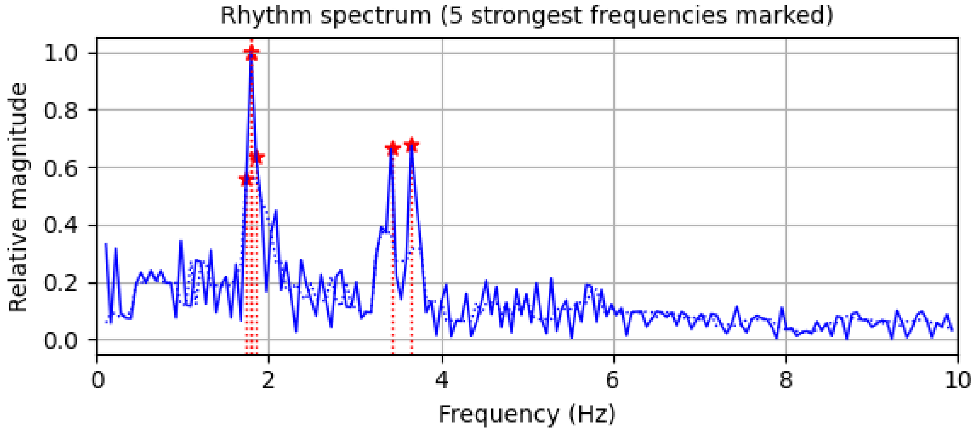
syllable rhythms
2.5 Hz
0.4 s

English: Counting to 30

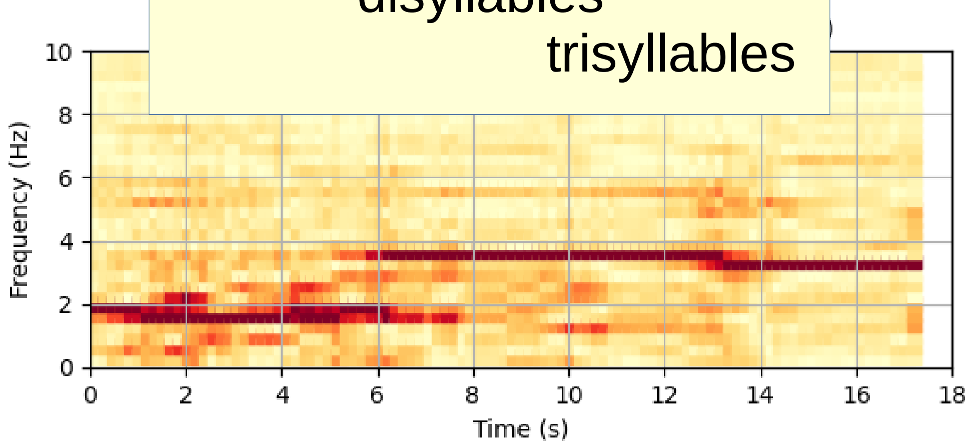
Waveform with smoothed positive amplitude envelope



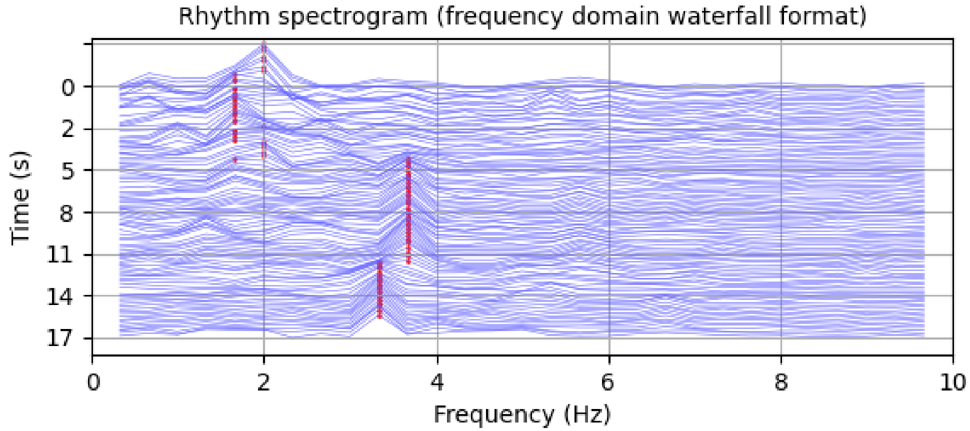
Rhythm spectrum (5 strongest frequencies marked)



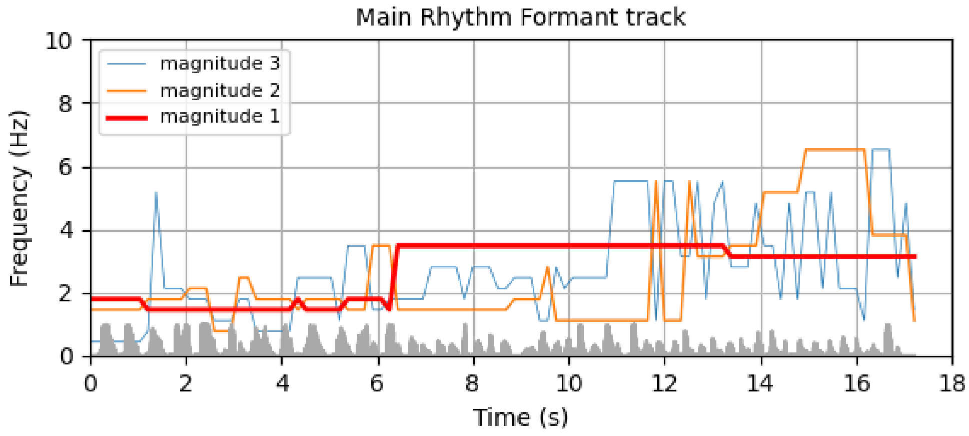
monosyllables
disyllables
trisyllables



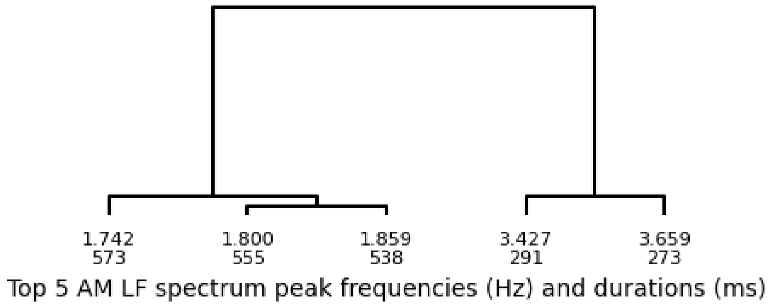
Rhythm spectrogram (frequency domain waterfall format)



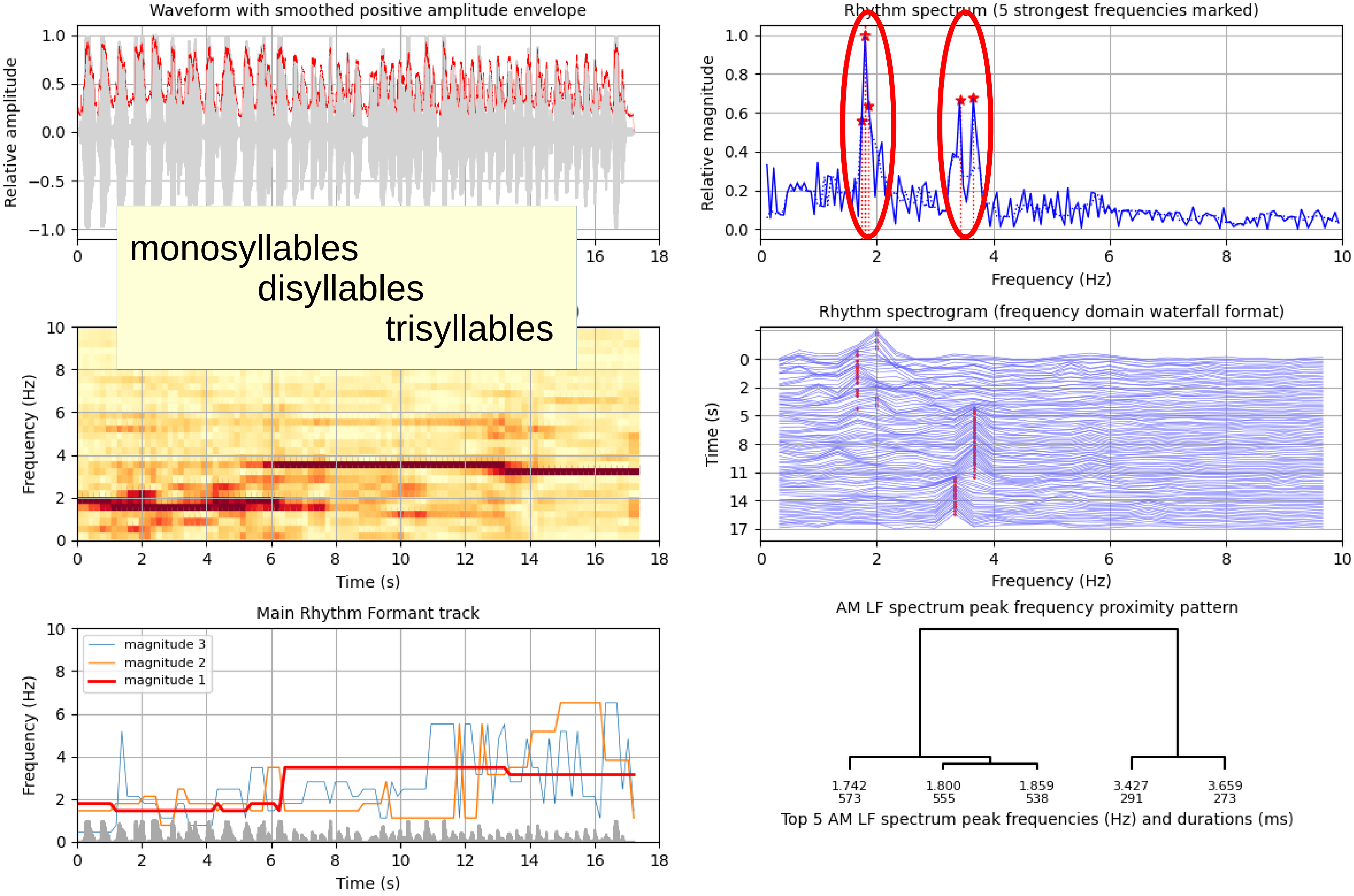
Main Rhythm Formant track



AM LF spectrum peak frequency proximity pattern

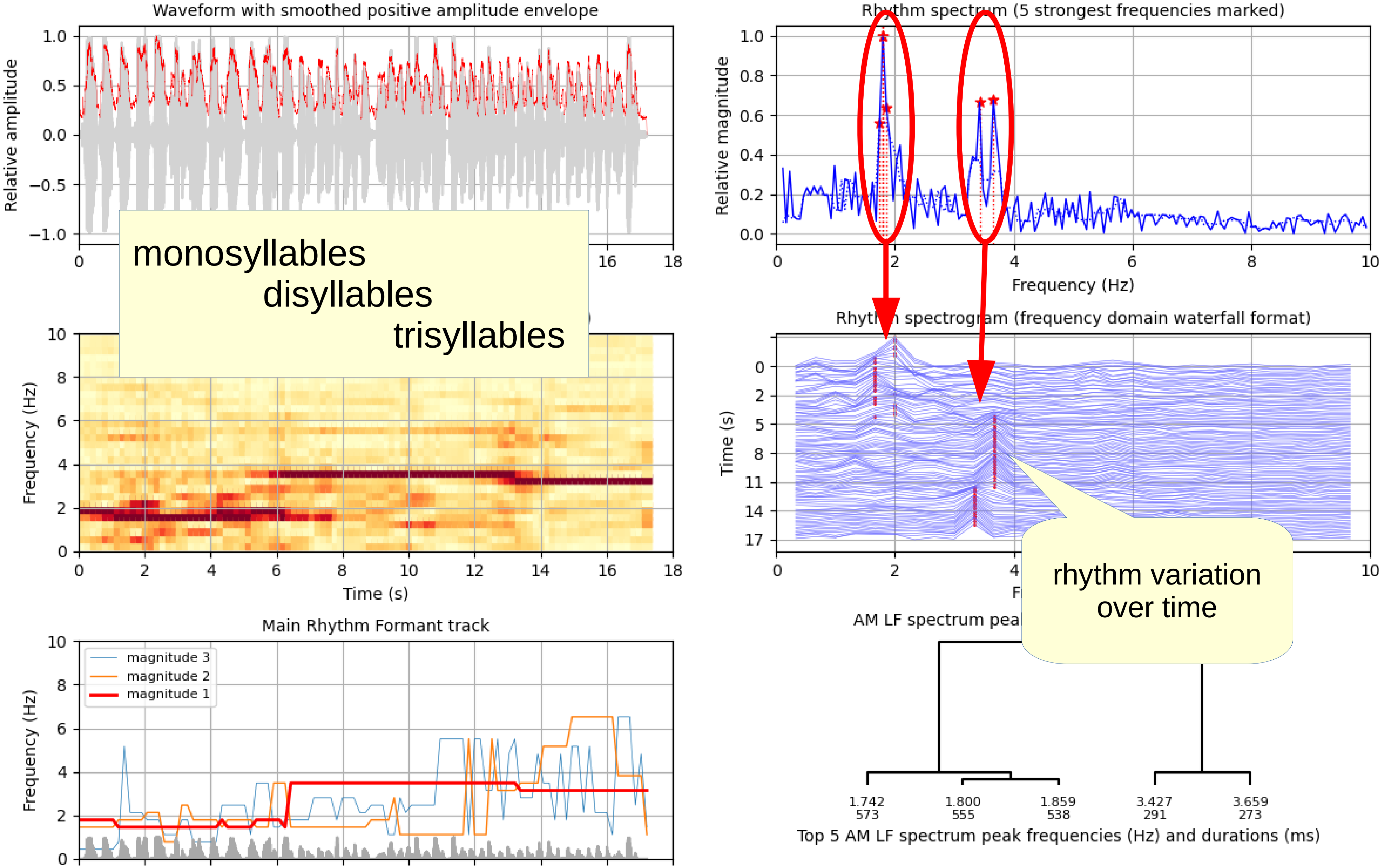

1.742
573
1.800
555
1.859
538
3.427
291
3.659
273
Top 5 AM LF spectrum peak frequencies (Hz) and durations (ms)

English: Counting to 30



monosyllables
disyllables
trisyllables

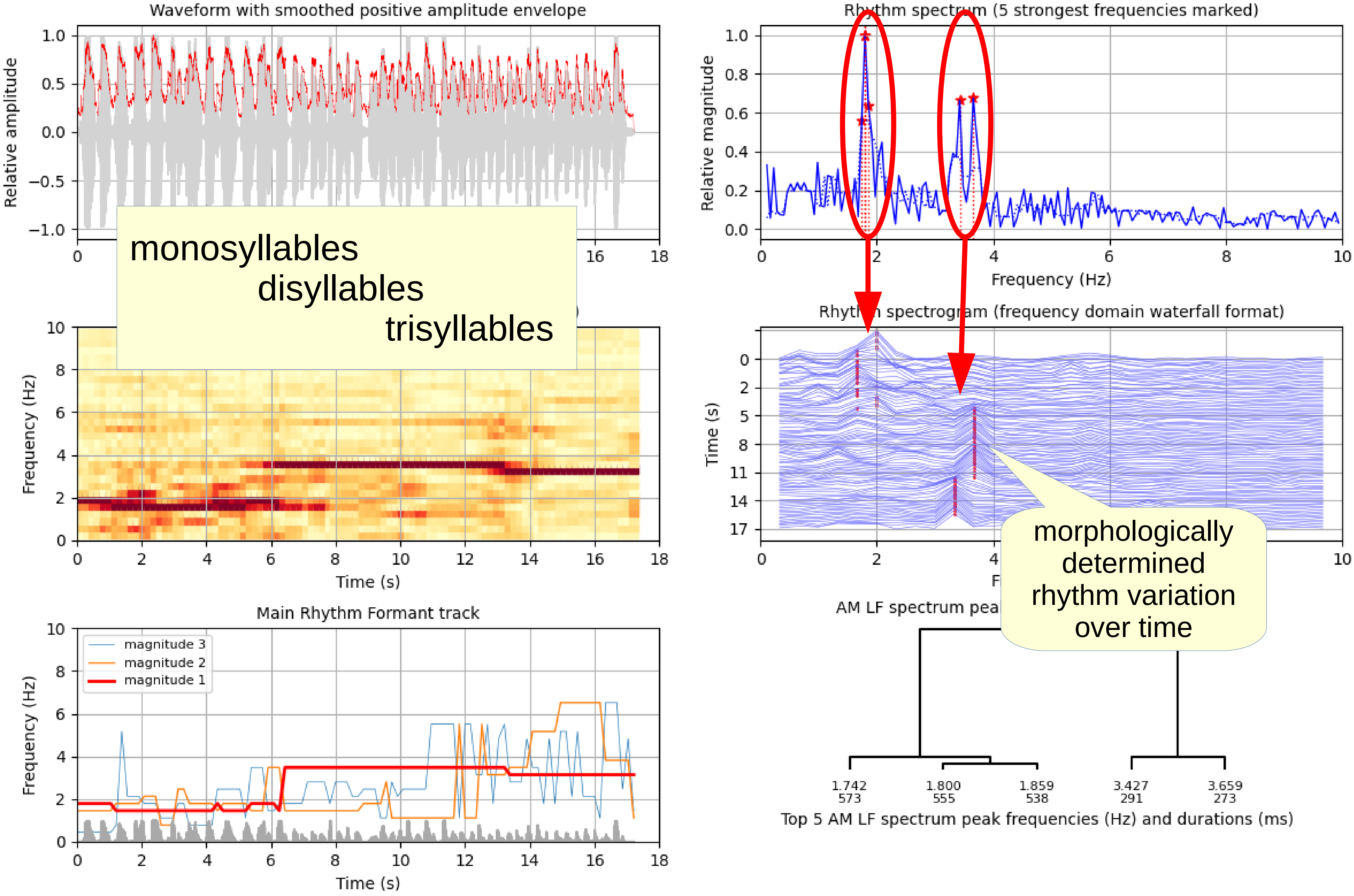
English: Counting to 30



monosyllables
disyllables
trisyllables

rhythm variation
over time

English: Counting to 30



monosyllables
disyllables
trisyllables

morphologically determined rhythm variation over time

Complexity of Prosodic Forms

Complexity of Prosodic Forms

Most approaches to prosody – even if they ‘look’ formal – use *informal* and *qualitative* descriptions.

Formal models, based on mathematics or logic, allow interesting properties such as the *complexity* of language and speech models to be defined.

Sometimes formal models are used informally, but this can lead to misunderstandings. A couple of cases will be shown in this lecture, for example in the case of misunderstandings about the concept of *recursion*.

Complexity of Prosodic Forms

1. Some current popular transcription systems are very local and atomistic:

   Individual pitch accents are transcribed independently of their neighbours (e.g. with the ToBI notation)

   This is an unjustified abstraction as Ladd pointed out over 30 years ago (1988)

2. Several important properties are ignored:
   1. similarity of pitch accents in sequences: (Type 3).
   2. different final pitch accent (Type 3):
      1. phonetic influence of boundary, final lengthening, etc.
      2. functional influence of (non-)termination, etc.
   3. different onset pitch accent (Type 3), pronounced height, range, contour, etc.
   4. global slope and prosodic hierarchy (declination, inclination, sustained)

3. How can an adequate model be obtained? Note:
   1. a hierarchy is not necessarily recursive
   2. some kinds of recursion are actually linear

Complexity of Prosodic Forms

There are different ways to define complexity in linguistics:

1. Complexity of *structures*, i.e. representations, for instance the number of nodes and connections in a network, the number of categories and rules in a grammar, the size of a search space and the number of constraints limiting it.

2. Complexity of *algorithms*, i.e. the functions relating the size of an input to the *time* or the memory *space* required for processing it.

The second case is particularly interesting:

In the 1950s, Chomsky and Schützenberger established a hierarchy of formal language types, each described by grammars with different time complexity.

Complexity of Prosodic Forms: Formal language hierarchy

The Chomsky-Schützenberger hierarchy of formal string languages:

Type 0: Unrestricted languages (with arbitrary connections over the string elements)

Type 1: Context-sensitive languages (with trees plus cross-links over the string elements)

Type 2: Context-free languages (with centre-embedding tree structures over the string elements)

Type 3: Regular (linear) languages (with right or left branching trees or linear links between string elements)

This set of language types is a hierarchy in the sense that there is a relation of inclusion between these languages:

$$\text{Type 3} \subset \text{Type 2} \subset \text{Type 1} \subset \text{Type 0}$$

It is important to understand this inclusion relation for an understanding of which model is the simplest model which is consistent with the facts.

Complexity of Prosodic Forms: Formal grammar hierarchy

The grammars which describe these languages and the automata which process them (omitting some important distinctions such as deterministic vs. nondeterministic grammars)

Type 0: Unrestricted grammars

- processing defined by Turing machines)
- time complexity exponential

Type 1: Context-sensitive grammars

- processing defined by linear-bounded automata
- time complexity theoretically polynomial, practically exponential

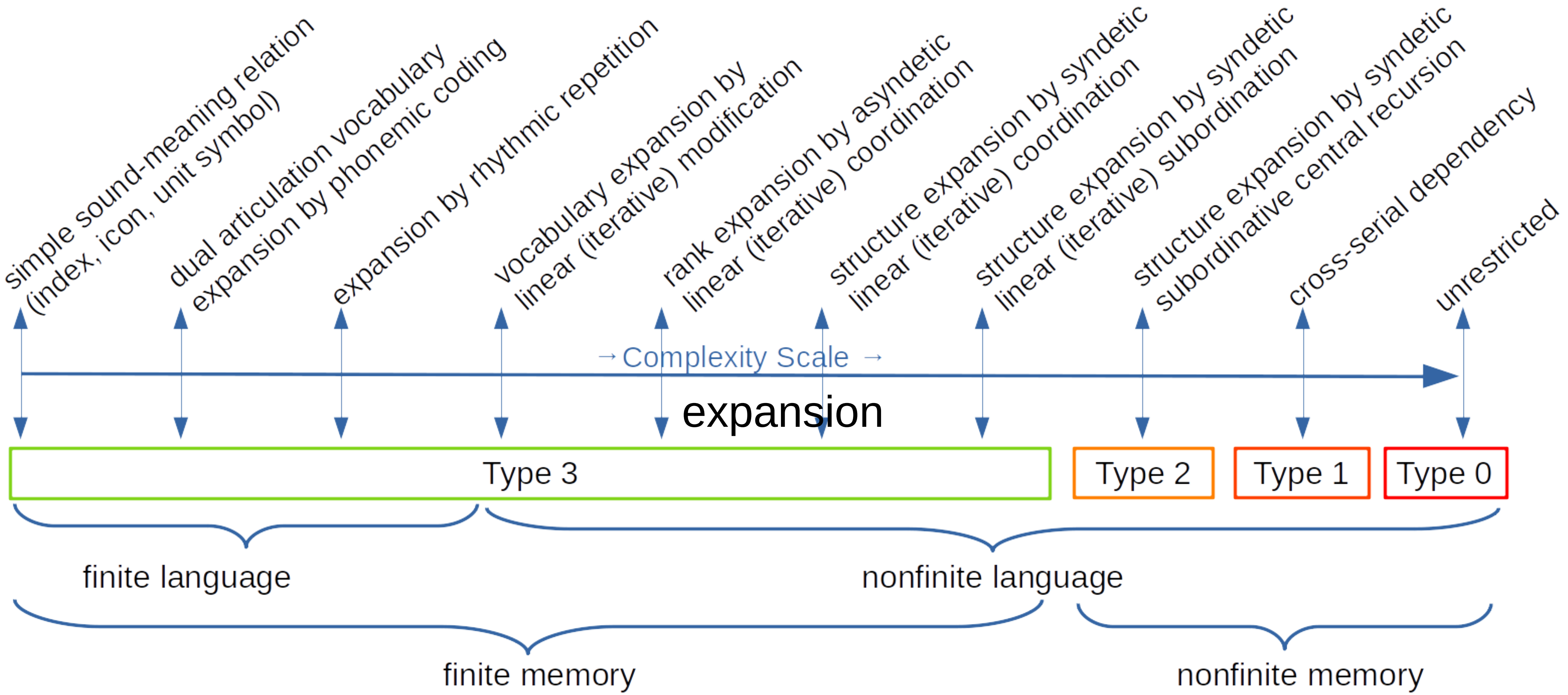
Type 2: Context-free (phrase structure) grammars

- processing defined by push-down automata
- time complexity sometimes handled as exponential but actually polynomial, in fact cubic (actually slightly less than cubic)

Type 3: Regular (linear) grammars

- processing defined by finite state automata
- time complexity linear if deterministic
- a regular grammar and its finite state automaton can always be made deterministic

Complexity of Prosodic Forms: Overview


simple sound-meaning relation (index, icon, unit symbol)
dual articulation vocabulary expansion by phonemic coding
expansion by rhythmic repetition
vocabulary expansion by linear (iterative) modification
rank expansion by asyndetic linear (iterative) coordination
structure expansion by syndetic linear (iterative) coordination
structure expansion by syndetic linear (iterative) subordination
structure expansion by syndetic subordinative central recursion
cross-serial dependency
unrestricted
→ Complexity Scale →
expansion
Type 3
Type 2
Type 1
Type 0
finite language
nonfinite language
finite memory
nonfinite memory

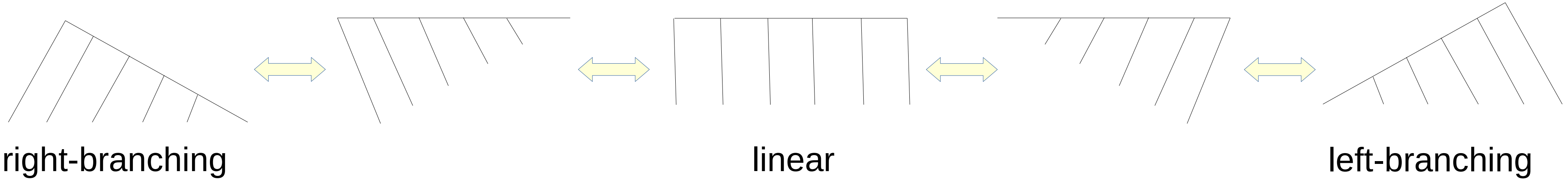
Chomsky-Schützenberger Hierarchy of Formal languages:

Type 3: left or right branching regular/linear grammar, finite state automaton
Type 2: context free (phrase structure) grammar, push-down automaton
Type 1: context-sensitive grammar, linear-bounded automaton
Type 0: unrestricted/transformational grammar, Turing machine

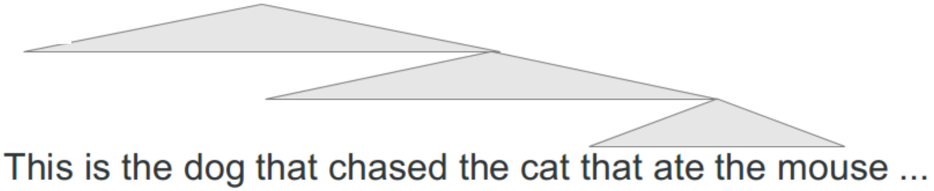
Investigating the Complexity Hierarchy

Investigating the Complexity Hierarchy

1. Recursion? One must ask: What kind of recursion?
    1. Work in the past 20 years in general does not distinguish between types of recursion.
    2. Note that a tree model of a hierarchy is defined recursively in graph-theoretic terms, but this does not necessarily mean <u>recursion</u> in a linguistic description:
        1. finite length forms are not recursive (e.g. syllables).
        2. finite depth forms are not recursive (e.g. the Strict Layer Hypothesis version of the Prosodic Hierarchy).
2. Right-recursive and left-recursive (right-branching and left-branching, tail-recursive and head-recursive) structures are equivalent to linear systems and are NOT centre-recursive (centre-embedding, self-embedding).
    1. In practice, the recursion is usually just right-branching, which is actually linear and is easily modelled by a finite state automaton
       (cf. earlier work by Fujisaki, ’t Hart, Pierrehumbert, Gibbon)
    2. So iteration, finite state automata, etc. seem to be sufficient to account for various intonational hierarchy effect.
    3. A simple illustration:



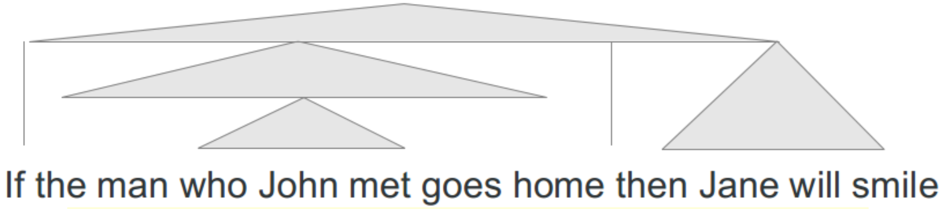
Complexity Hierarchy, Structure and Prosody in Practice



Right-branching linear recursion / iteration.

Type 3: Regular Grammar

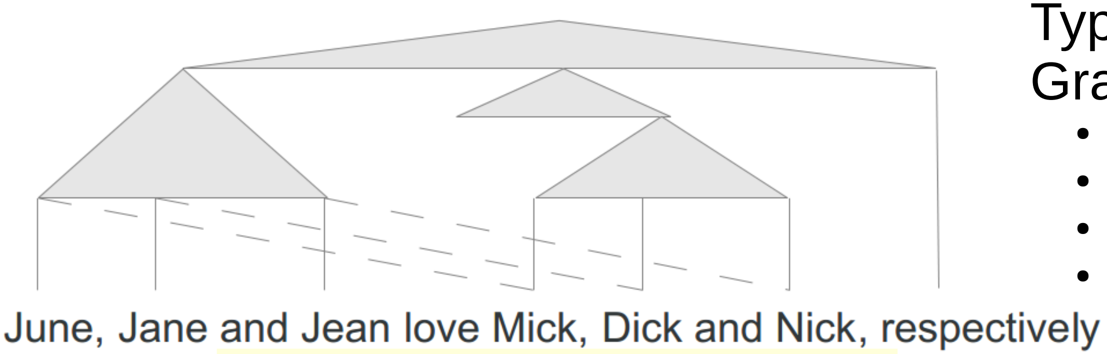
- left or right branching → iteration
- linear processing time
- finite memory
- processed by finite state automaton



Centre-embedding hierarchical recursion.

Type 2: Context-free (Phrase Structure) Grammar

- centre-embedding
- near-cubic polynomial processing time
- non-finite memory
- processed by push-down automaton



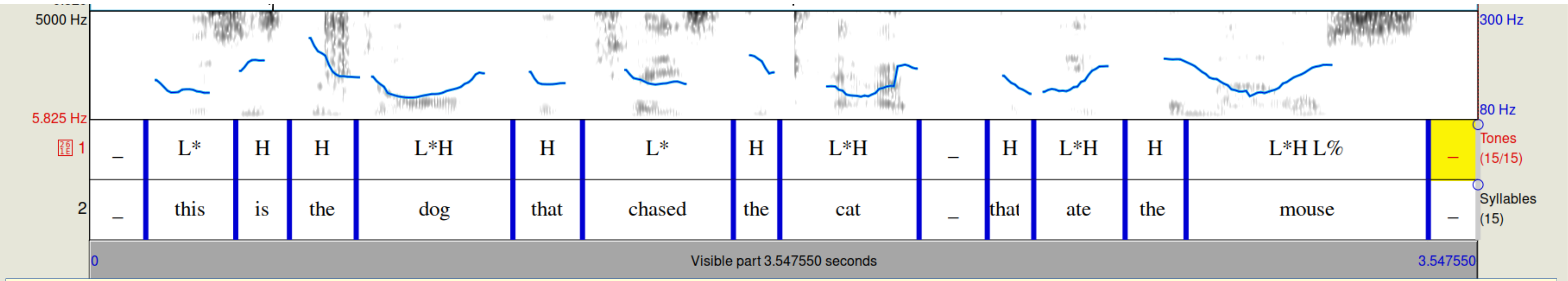
Recursive cross-serial dependency.

Type 1: Context-sensitive & Indexed Grammars

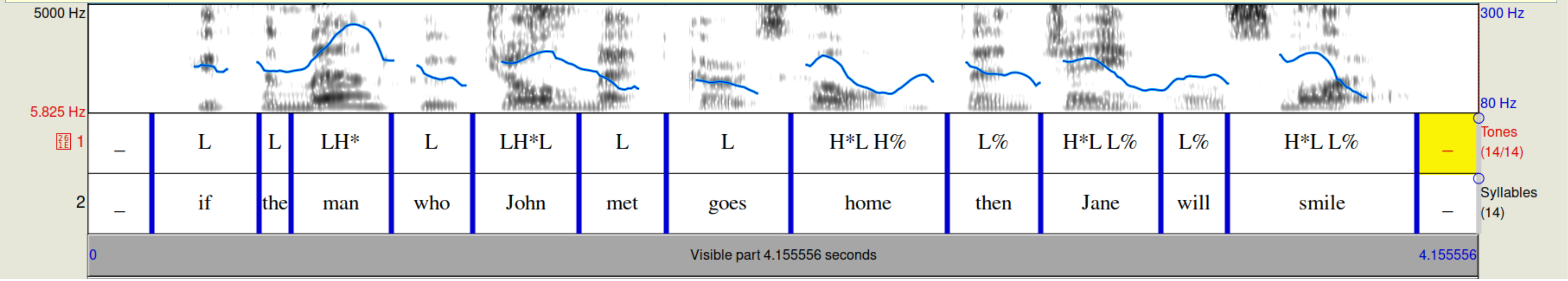
- cross-linked branching
- up to exponential processing time
- non-finite memory
- processed by linear-bounded automaton

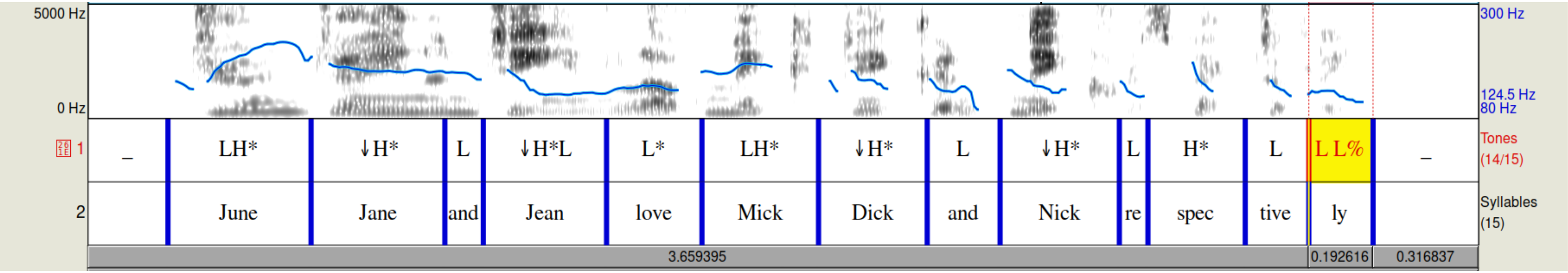
So let’s take a look at some examples.

Complexity Hierarchy, Structure and Prosody in Practice

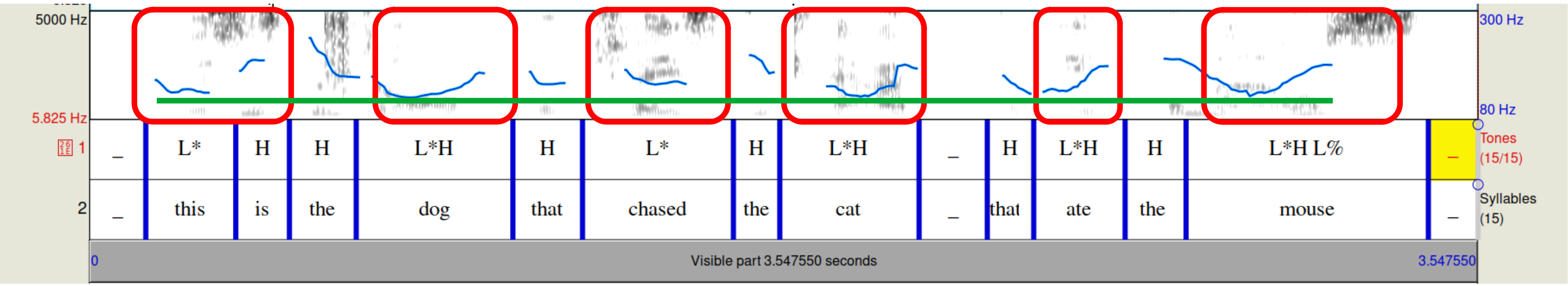

5000 Hz
5.825 Hz
300 Hz
80 Hz
1
_ L* H H L*H H L* H L*H _ H L*H H L*H L% _
Tones
(15/15)
2
_ this is the dog that chased the cat _ that ate the mouse _
Syllables
(15)
0
Visible part 3.547550 seconds
3.547550

There are many options for pitch contours in English, however complex the syntax


5000 Hz
5.825 Hz
300 Hz
80 Hz
1
_ L L LH* L LH*L L L H*L H% L% H*L L% L% H*L L% _
Tones
(14/14)
2
_ if the man who John met goes home then Jane will smile _
Syllables
(14)
0
Visible part 4.155556 seconds
4.155556


5000 Hz
0 Hz
300 Hz
124.5 Hz
80 Hz
1
_ LH* ↓H* L ↓H*L L* LH* ↓H* L ↓H* L H* L L L% _
Tones
(14/15)
2
June Jane and Jean love Mick Dick and Nick re spec tive ly
Syllables
(15)
3.659395
0.192616
0.316837

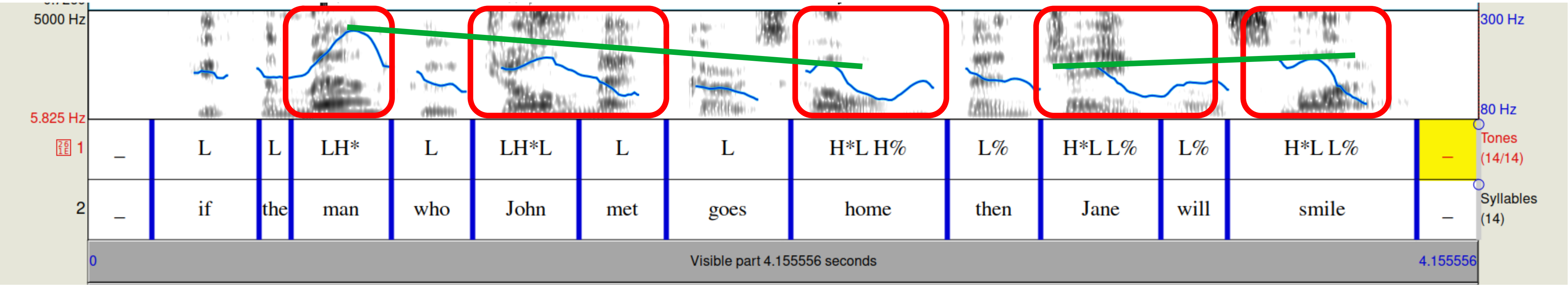
Prosodic Complexity? The Chomsky-Schützenberger Hierarchy



1	_	L*	H	H	L*H	H	L*	H	L*H	_	H	L*H	H	L*H L%	_
2	_	this	is	the	dog	that	chased	the	cat	_	that	ate	the	mouse	_

Tones (15/15)
Syllables (15)

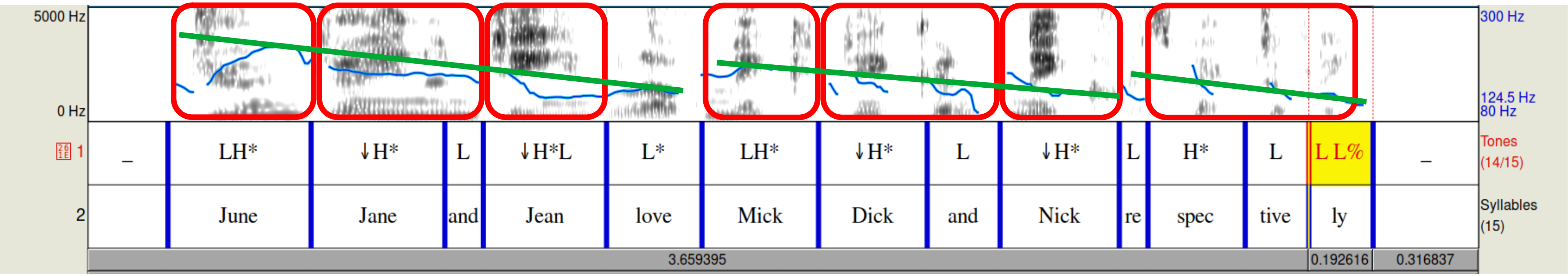
0 Visible part 3.547550 seconds 3.547550



1	_	L	L	LH*	L	LH*L	L	L	H*L H%	L%	H*L L%	L%	H*L L%	_
2	_	if	the	man	who	John	met	goes	home	then	Jane	will	smile	_

Tones (14/14)
Syllables (14)

0 Visible part 4.155556 seconds 4.155556



1	_	LH*	↓H*	L	↓H*L	L*	LH*	↓H*	L	↓H*	L	H*	L	L L%	_
2		June	Jane	and	Jean	love	Mick	Dick	and	Nick	re	spec	tive	ly	

Tones (14/15)
Syllables (15)

3.659395 0.192616 0.316837

But it all comes back to Type 3 (regular, linear) grammars

Left-branching and right-branching grammars are Type 3 (and by implication also Type 2 etc.).

Unlike strictly Type 2 (context-free, phrase structure) languages, the Type 3 languages are *NOT* centre-embedding (‘self-embedding’) but left or right recursive (head or tail recursive):

1. For each left-branching (left-recursive, head recursive) Type 3 grammar there is a weakly equivalent right-branching (right-recursive, tail recursive) Type 3 grammar and vice versa (i.e. a grammar which generates the same language).
2. Every Type 3 grammar can be converted into a weakly equivalent finite state automaton as a transition table or transition network (FSA, FSN) and vice versa.

   In particular, every <u>head-recursive</u> or <u>tail-recursive</u> Type 3 grammar is weakly equivalent to an <u>iterative</u> finite state automaton, i.e. an automaton with ‘loops’.

Example – a very small but infinite subset of English:

L = { *it is good, it is very good, it is very very good, it is very very very good, ...* }

Right-branching Type 3:
A → it B
B → is C
C → very C
C → good

Left-branching Type 3:
A → B good
B → B very
B → C is
C → it

But it all comes back to Type 3 (regular, linear) grammars

Left-branching and right-branching grammars are Type 3 (and by implication also Type 2 etc.).

Unlike strictly Type 2 (context-free, phrase structure) languages, the Type 3 languages are *NOT* centre-embedding (‘self-embedding’) but left or right recursive (head or tail recursive):

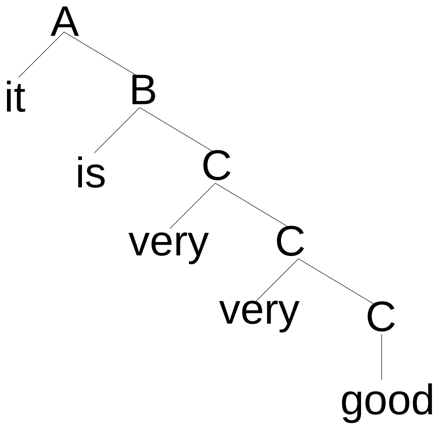
1. For each left-branching (left-recursive, head recursive) Type 3 grammar there is a weakly equivalent right-branching (right-recursive, tail recursive) Type 3 grammar and vice versa (i.e. a grammar which generates the same language).
2. Every Type 3 grammar can be converted into a weakly equivalent finite state automaton as a transition table or transition network (FSA, FSN) and vice versa.
   In particular, every <u>head-recursive</u> or <u>tail-recursive</u> Type 3 grammar is weakly equivalent to an <u>iterative</u> finite state automaton, i.e. an automaton with ‘loops’.

Example – a very small but infinite subset of English:

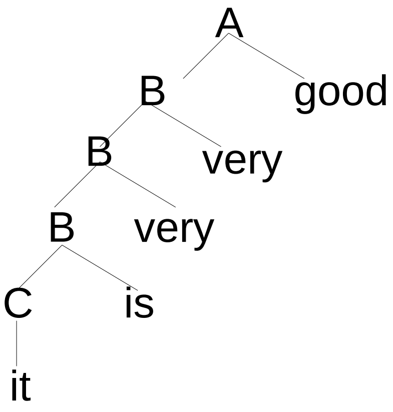
L = { *it is good, it is very good, it is very very good, it is very very very good, ...* }

Right-branching Type 3:
A → it B
B → is C
C → very C
C → good

Left-branching Type 3:
A → B good
B → B very
B → C is
C → it



right-branching



left-branching

But it all comes back to Type 3 (regular, linear) grammars

Left-branching and right-branching grammars are Type 3 (and by implication also Type 2 etc.).

Unlike strictly Type 2 (context-free, phrase structure) languages, the Type 3 languages are *NOT* centre-embedding (‘self-embedding’) but left or right recursive (head or tail recursive):

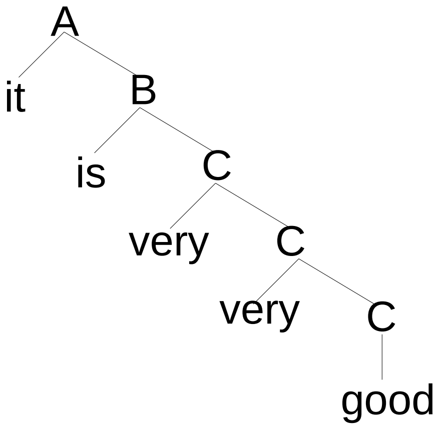
1. For each left-branching (left-recursive, head recursive) Type 3 grammar there is a weakly equivalent right-branching (right-recursive, tail recursive) Type 3 grammar and vice versa (i.e. a grammar which generates the same language).
2. Every Type 3 grammar can be converted into a weakly equivalent finite state automaton as a transition table or transition network (FSA, FSN) and vice versa.
   In particular, every <u>head-recursive</u> or <u>tail-recursive</u> Type 3 grammar is weakly equivalent to an <u>iterative</u> finite state automaton, i.e. an automaton with ‘loops’.

Example – a very small but infinite subset of English:

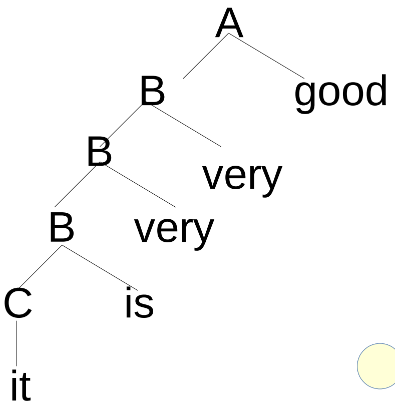
L = { *it is good, it is very good, it is very very good, it is very very very good, ...* }

Right-branching Type 3:
A → it B
B → is C
C → very C
C → good

Left-branching Type 3:
A → B good
B → B very
B → C is
C → it

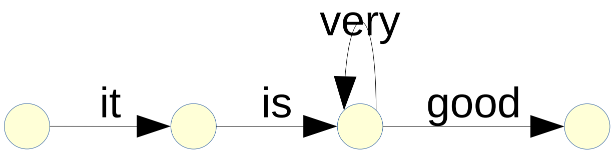


right-branching



left-branching

Right-recursive grammars are equivalent to iterative finite state automata.



FSN ≡ FSA

But it all comes back to Type 3 (regular, linear) grammars

Why are Type 3 languages and grammars important?

They are weakly equivalent to Finite State Automata.
An FSA only requires

- linear time (real time) in relation to the length of the input
- finite memory in relation to the size of the grammar

In contrast, Types 0...2 require

- polynomial or exponential time in relation to the length of the input
- non-finite memory

This is an over-generalisation and unsuitable as a model of human processing

Why are these equivalences important?

Many constituents of languages are right-branching. Therefore their grammar can be converted into a weakly equivalent iterative FSA.

In the 1980s it was established that

1. intonation patterns (Pierrehumbert 1980; Gibbon 1984) and
2. tonal patterns (Gibbon 1987, Niger-Congo tone languages; Jansche 1997, Tianjin Mandarin)

can be modelled with FSAs.

So a mapping between a right-branching constructions and an intonation pattern is not necessarily based on centre-embedding but on pairs of linear structures..

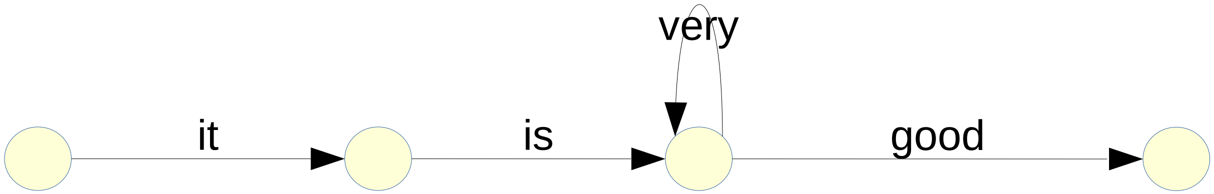
But it all comes back to Type 3 (regular, linear) grammars

From Finite State Automata to Finite State Transducers

As shown by Koskenniemi (****), Kaplan & Kay (1994); Beesley & Karttunen (2003); Gibbon (1987, 2001), mappings between linear sequences in morphology, phonology and prosody can be represented by a finite state transducer (FST).

A Finite State Transducer operates over strings of pairs (or triples, larger tuples etc.) rather than strings of single elements and is bidirectional.

Like a (much too) simple reversible translator:



English: it is very very ... good

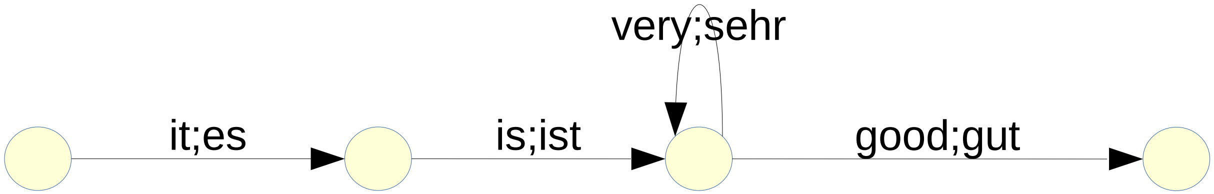
But it all comes back to Type 3 (regular, linear) grammars

From Finite State Automata to Finite State Transducers

As shown by Koskenniemi (****), Kaplan & Kay (1994); Beesley & Karttunen (2003); Gibbon (1987, 2001), mappings between linear sequences in morphology, phonology and prosody can be represented by a finite state transducer (FST).

A Finite State Transducer operates over strings of pairs (or triples, larger tuples etc.) rather than strings of single elements and is bidirectional.

Like a (much too) simple reversible translator:



English: it is very very ... good

German: es ist sehr sehr ... gut

From Finite State Automata to Finite State Transducers

By the way, note the origins of parallel phonologies and morphologies in the 1980s (Karttunen 2012):

"... Koskenniemi invented a new way to describe phonological alternations in finite-state terms. Instead of cascaded rules with intermediate stages and the computational problems they seemed to lead to, rules could be thought of as statements that directly constrain the surface realization of lexical strings. The rules would not be applied sequentially but in parallel. Each rule would constrain a certain lexical/surface correspondence and the environment in which the correspondence was allowed, required, or prohibited. For his 1983 dissertation, Koskenniemi constructed an ingenious implementation of his constraint-based model that did not depend on a rule compiler, composition or any other finite-state algorithm, and he called it TWO-LEVEL MORPHOLOGY."

From Finite State Automata to Finite State Transducers

By the way, note the origins of parallel phonologies and morphologies in the 1980s (Karttunen 2012):

"... Koskenniemi invented a new way to describe phonological alternations in finite-state terms. Instead of cascaded rules with intermediate stages and the computational problems they seemed to lead to, rules could be thought of as statements that directly constrain the surface realization of lexical strings. The rules would not be applied sequentially but in parallel. Each rule would constrain a certain lexical/surface correspondence and the environment in which the correspondence was allowed, required, or prohibited. For his 1983 dissertation, Koskenniemi constructed an ingenious implementation of his constraint-based model that did not depend on a rule compiler, composition or any other finite-state algorithm, and he called it TWO-LEVEL MORPHOLOGY."

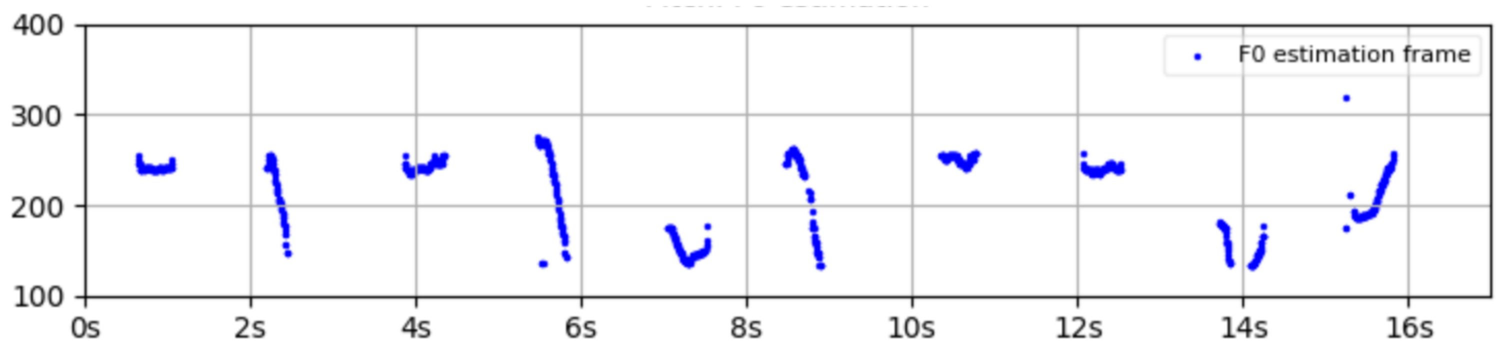
Does this sound like Optimality Theory? It's not an accident.

Finite State Intonation Models

Tones, Pitch Accents and Intonation: the ‘Modulation Code’

Sino-Tibetan
Pǔtōnghuà
ISO-693-3 *cmn*

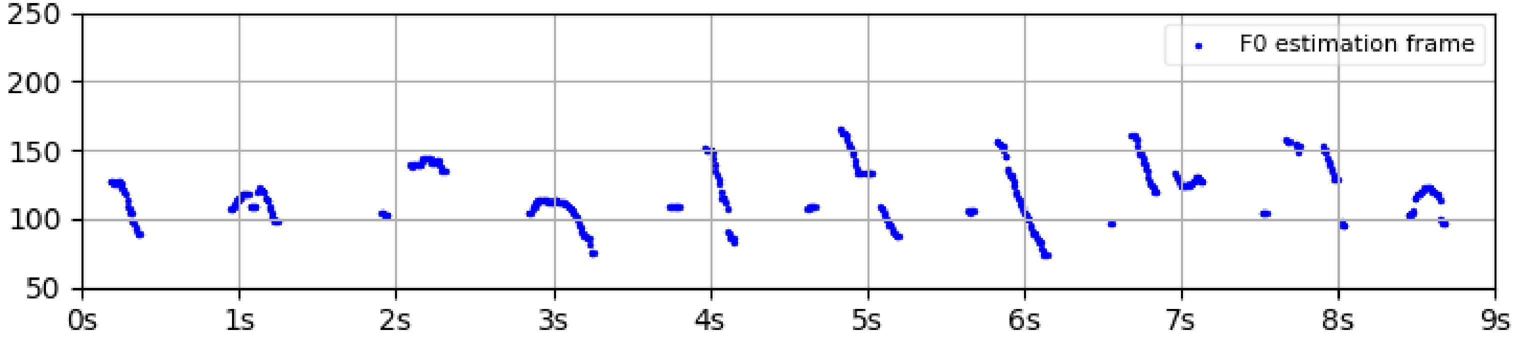
lexical tone



yī èr sān sì wǔ liù qī bā jiǔ shí

Niger-Congo
Ibibio
ISO-693-3 *ibb*

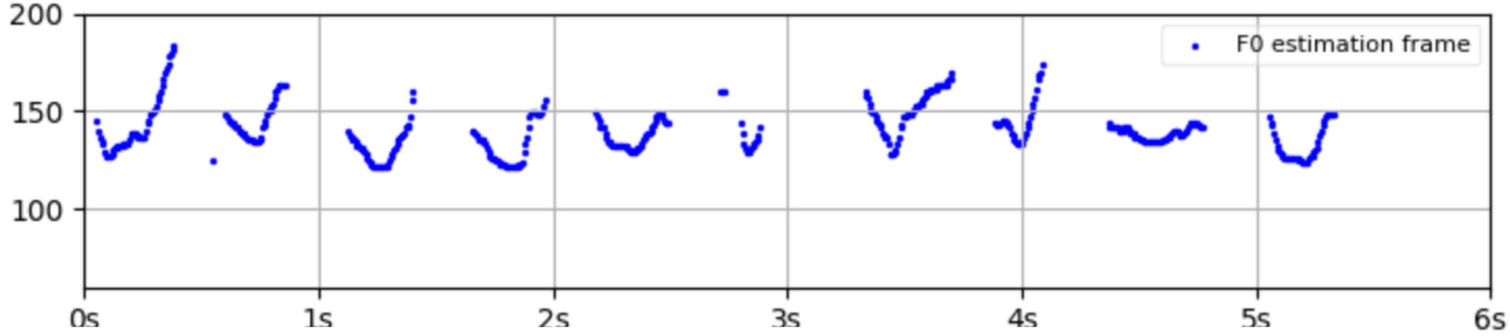
lexical and morphological tone



kèèd ìbà ìtá ìnààñ ìtíòn ìtíòkèèd ìtíàbà ìtiáìtá ùsúkkéèd dùòp

Indo-Germanic
English
ISO 693-3 *eng*

stress-pitch accent & intonation

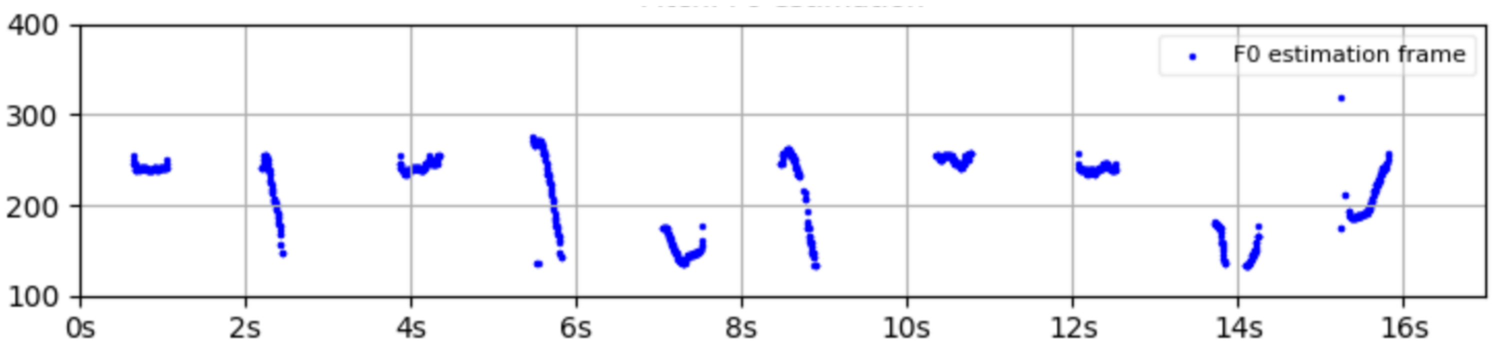


one two three four five six seven eight nine ten

Tones, Pitch Accents and Intonation: the ‘Modulation Code’

Sino-Tibetan
Pŭtōnghuà
ISO-693-3 *cmn*

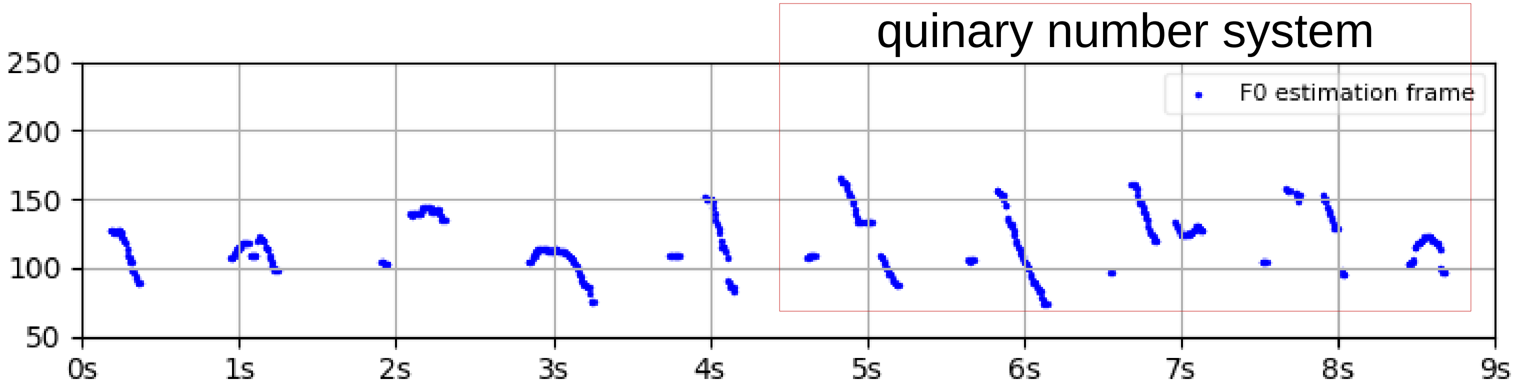
lexical tone



yī èr sān sì wŭ liù qī bā jiŭ shí

Niger-Congo
Ibibio
ISO-693-3 *ibb*

lexical and morphological tone

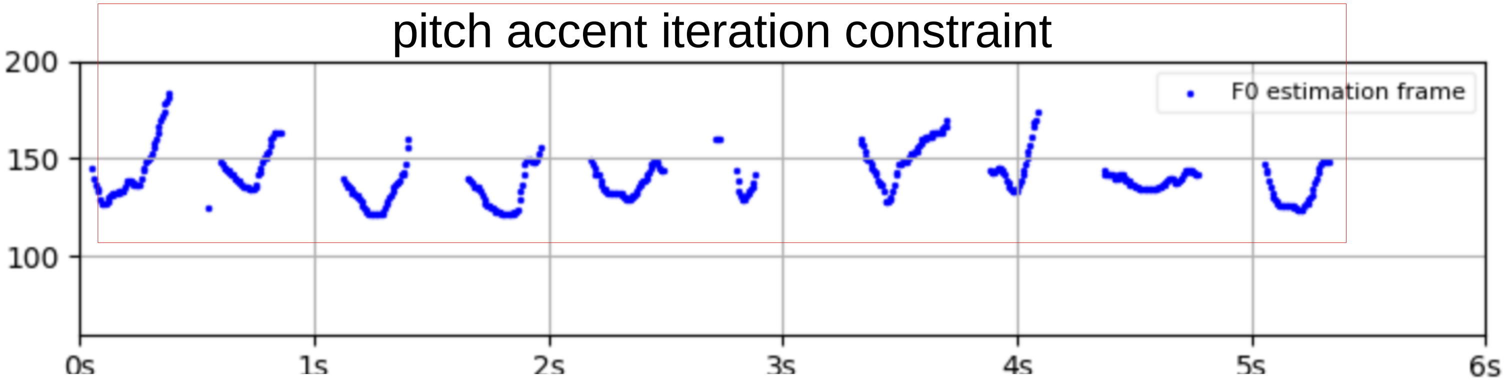


quinary number system

kèèd ìbà ìtá ìnààñ ìtíòn ìtíòkèèd ìtíàbà ìtiáìtá ùsúkkéèd dùòp

Indo-Germanic
English
ISO 693-3 *eng*

stress-pitch accent & intonation

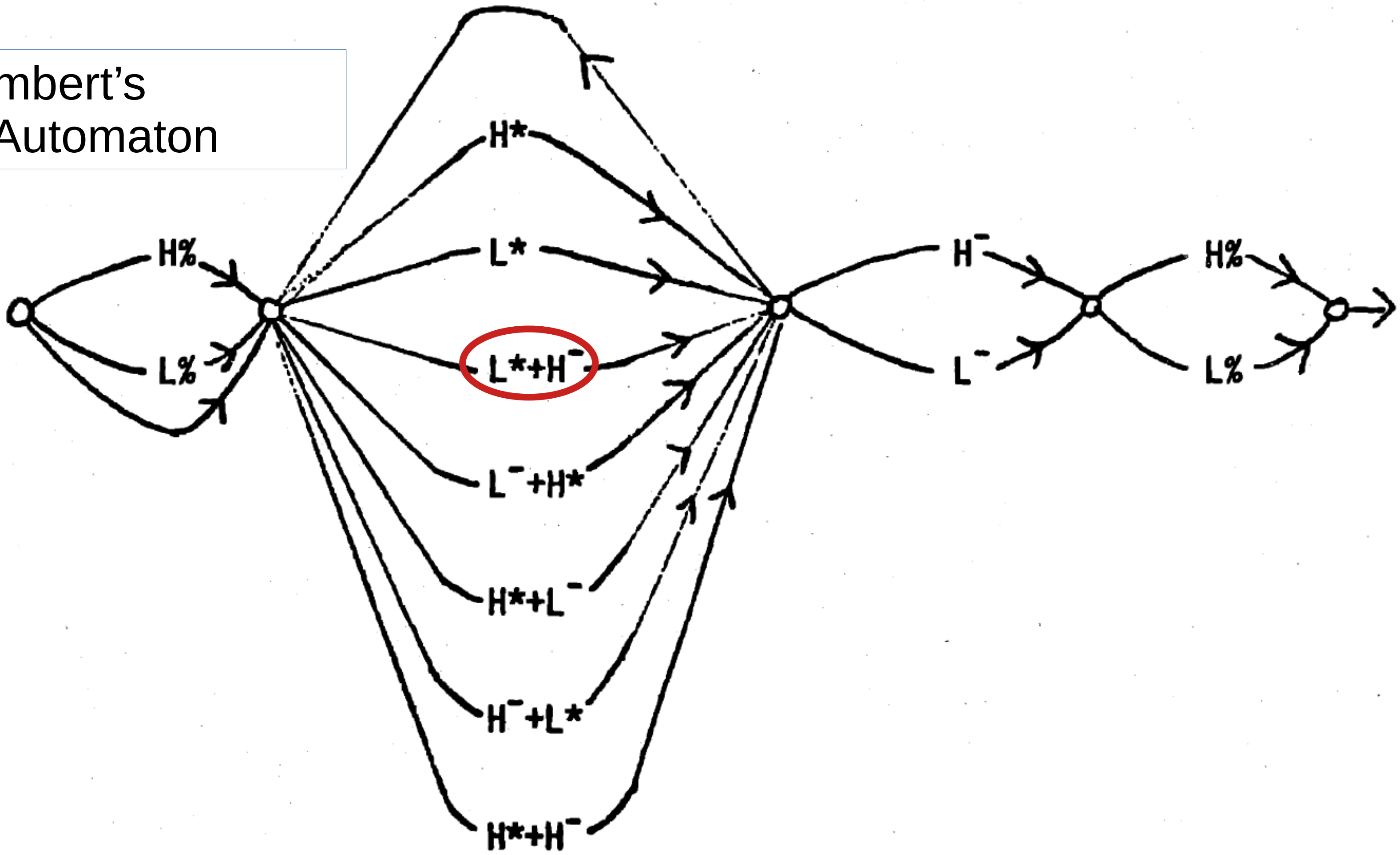


pitch accent iteration constraint

one two three four five six seven eight nine ten

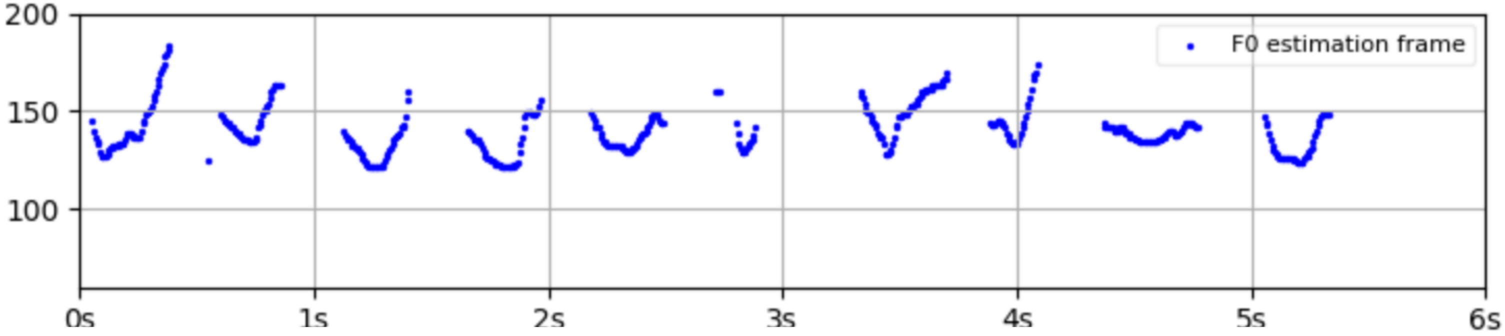
Tones, Pitch Accents and Intonation: the ‘Modulation Code’

Pierrehumbert’s
Finite State Automaton



Indo-Germanic
English
ISO 693-3 *eng*

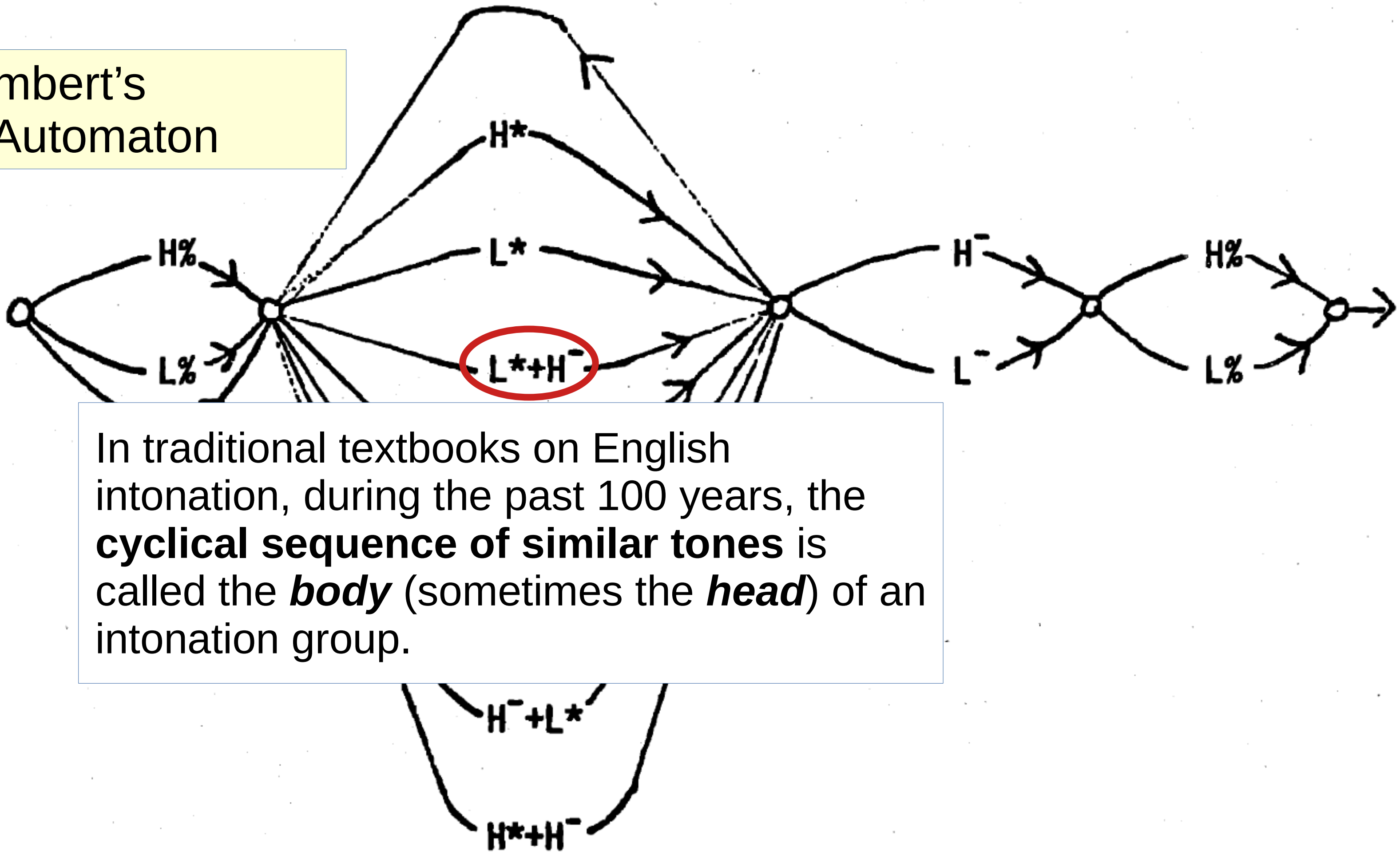
stress-pitch accent & intonation



one two three four five six seven eight nine ten

Tones, Pitch Accents and Intonation: the ‘Modulation Code’

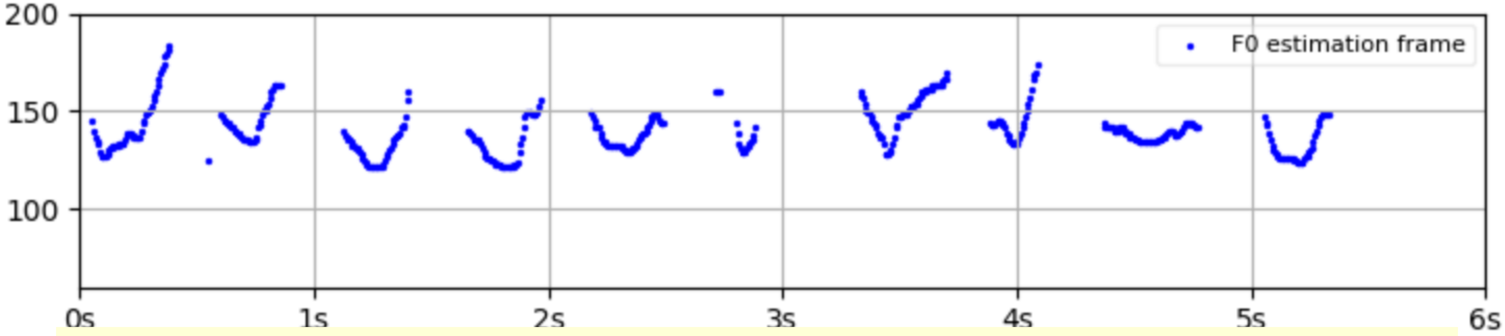
Pierrehumbert’s
Finite State Automaton



In traditional textbooks on English intonation, during the past 100 years, the **cyclical sequence of similar tones** is called the ***body*** (sometimes the ***head***) of an intonation group.

Indo-Germanic
English
ISO 693-3 *eng*

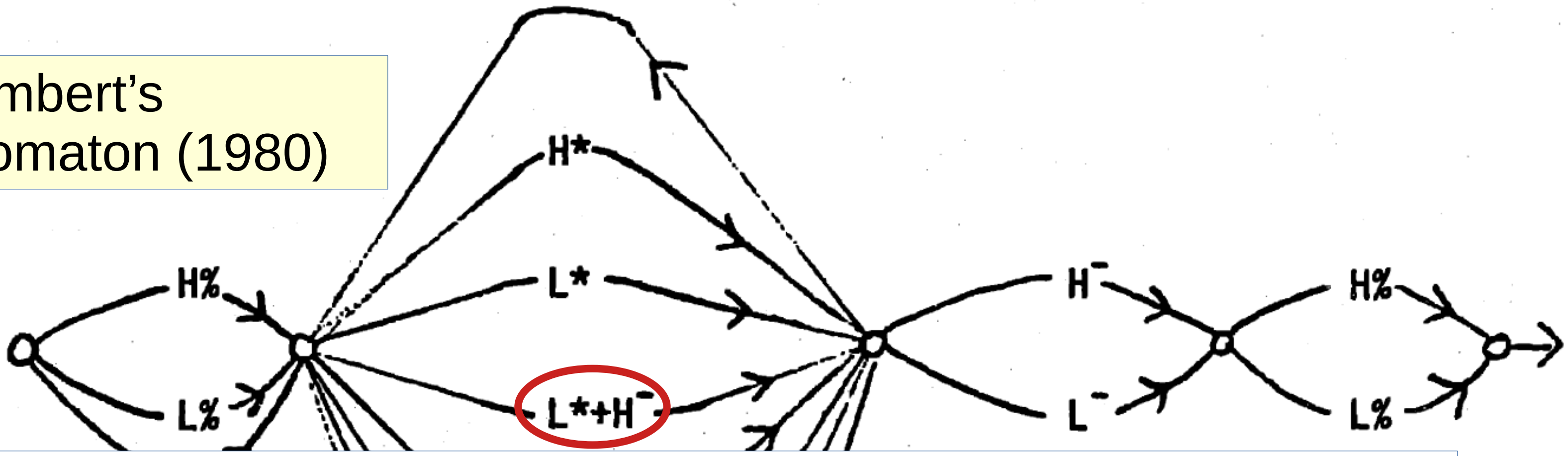
stress-pitch accent & intonation



one two three four five six seven eight nine ten

Tones, Pitch Accents and Intonation: the ‘Modulation Code’

Pierrehumbert’s
Finite State Automaton (1980)

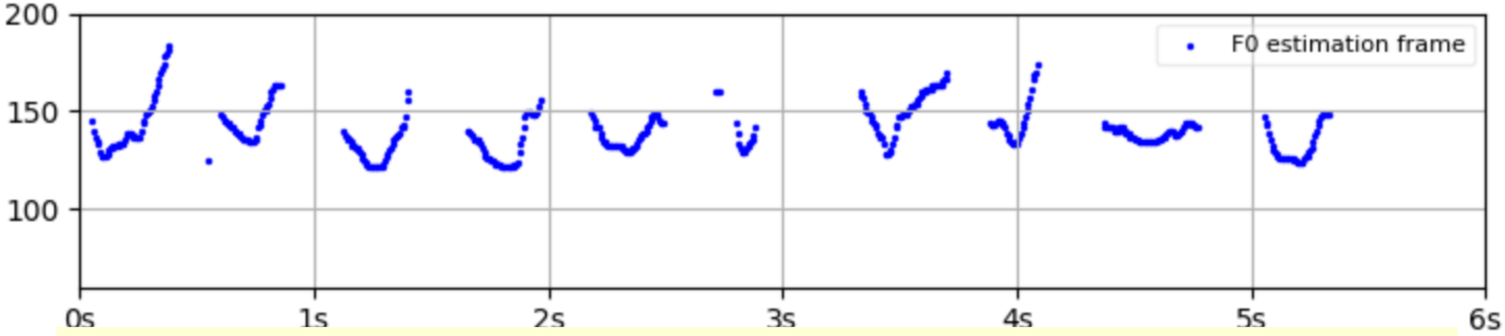


Dilley (1997: 87ff.)

- proposed an **accent sequence similarity constraint** for the head pattern,
- in order to explain such sequential pitch accent patterns as **correlate of coherent grammatical patterns** and
- as a means of **entraining the attention of listeners** to expect pattern changes such as nuclear tones.

Indo-Germanic
English
ISO 693-3 *eng*

stress-pitch accent & intonation



one two three four five six seven eight nine ten

Models of f0 patterning: Liberman & Pierrehumbert

Subtract the reference line from the F0 trajectory

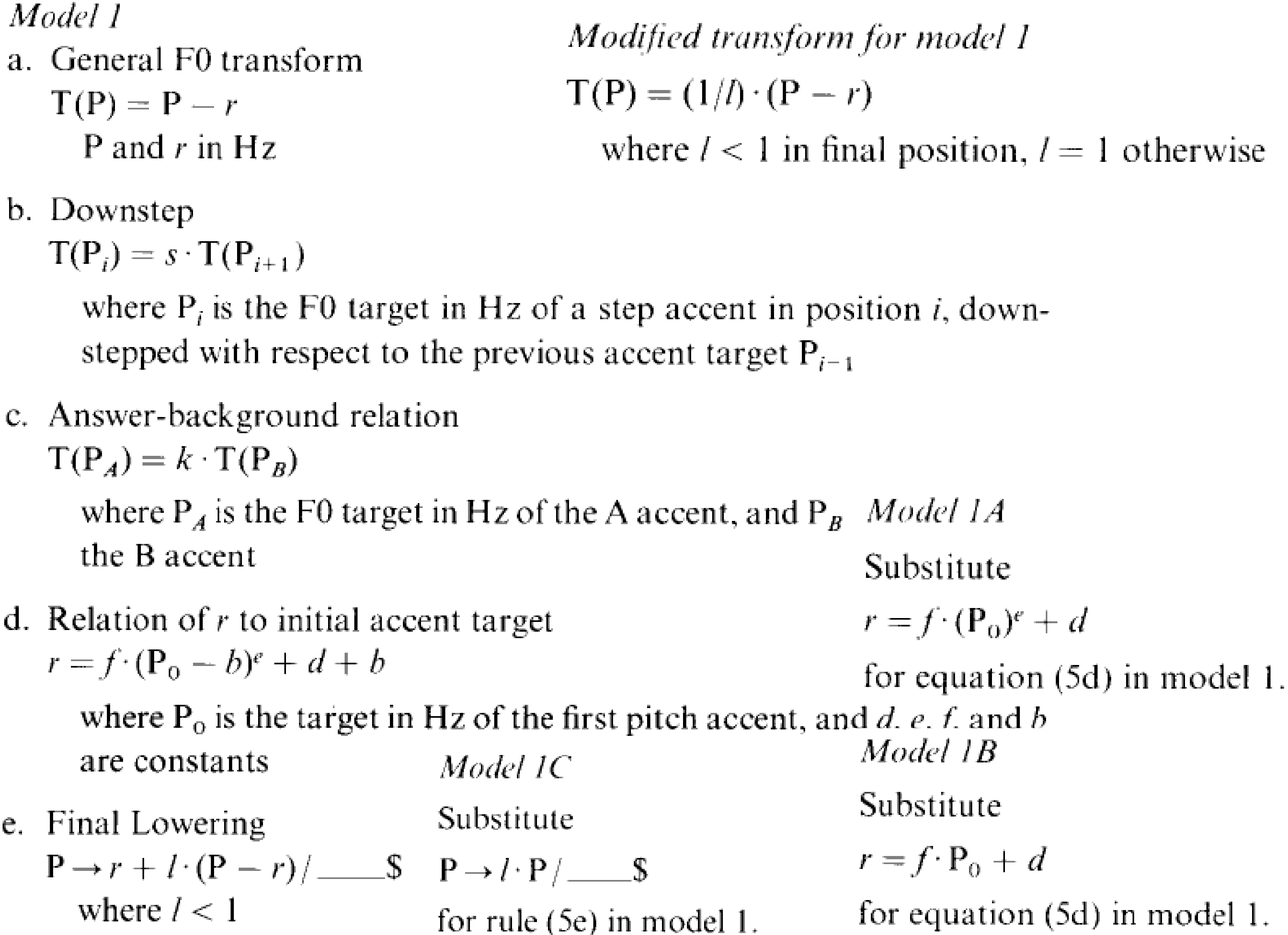
Define the asymptotic declination line

Define the relation between focus and non-focus accent types

Define the relation between first pitch accent and reference line

Define final lowering

Models of f0 patterning: Liberman & Pierrehumbert



Model 1

a. General F0 transform

$T(P) = P - r$

P and r in Hz

b. Downstep

$T(P_i) = s \cdot T(P_{i+1})$

where P_i is the F0 target in Hz of a step accent in position i, downstepped with respect to the previous accent target P_{i-1}

c. Answer-background relation

$T(P_A) = k \cdot T(P_B)$

where P_A is the F0 target in Hz of the A accent, and P_B the B accent

d. Relation of r to initial accent target

$r = f \cdot (P_0 - b)^e + d + b$

where P_0 is the target in Hz of the first pitch accent, and d. e. f. and b are constants

e. Final Lowering

$P \rightarrow r + l \cdot (P - r) / \_\_\_\_ \$$

where $l < 1$

Modified transform for model 1

$T(P) = (1/l) \cdot (P - r)$

where $l < 1$ in final position, $l = 1$ otherwise

Model 1A

Substitute

$r = f \cdot (P_0)^e + d$

for equation (5d) in model 1.

Model 1B

Substitute

$r = f \cdot P_0 + d$

for equation (5d) in model 1.

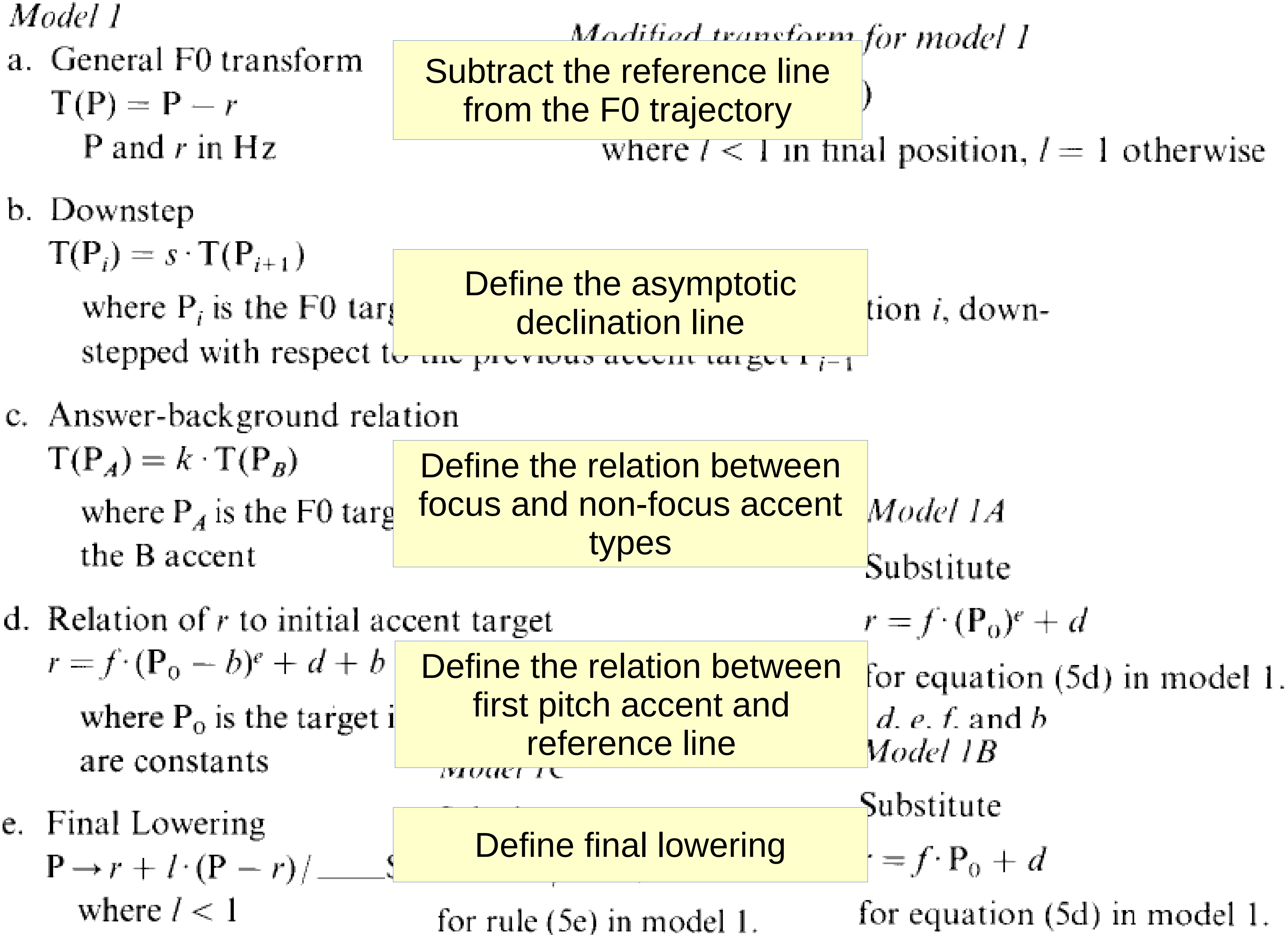
Model 1C

Substitute

$P \rightarrow l \cdot P / \_\_\_\_ \$$

for rule (5e) in model 1.

Models of f0 patterning: Liberman & Pierrehumbert



Model 1

a. General F0 transform

$T(P) = P - r$

P and r in Hz

Subtract the reference line from the F0 trajectory

b. Downstep

$T(P_i) = s \cdot T(P_{i+1})$

where P_i is the F0 targ tion i, down-stepped with respect to the previous accent target P_{i-1}

Define the asymptotic declination line

c. Answer-background relation

$T(P_A) = k \cdot T(P_B)$

where P_A is the F0 targ the B accent

Define the relation between focus and non-focus accent types

d. Relation of r to initial accent target

$r = f \cdot (P_0 - b)^e + d + b$

where P_0 is the target i are constants

Define the relation between first pitch accent and reference line

e. Final Lowering

$P \rightarrow r + l \cdot (P - r) / _____$S

where $l < 1$

Define final lowering

Modified transform for model 1

)

where $l < 1$ in final position, $l = 1$ otherwise

Model 1A

Substitute

$r = f \cdot (P_0)^e + d$

for equation (5d) in model 1.

d, e, f, and b

Model 1B

Substitute

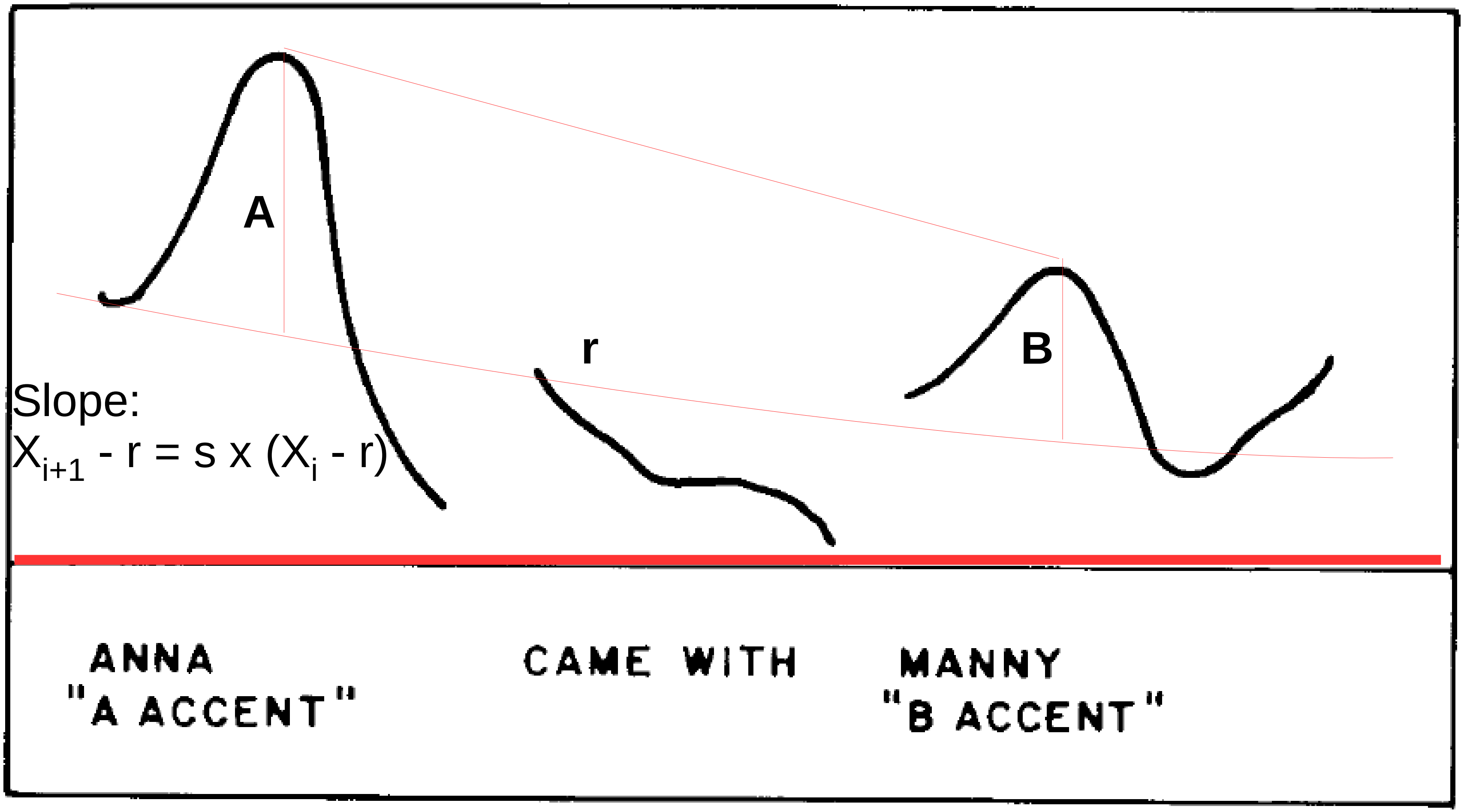
$r = f \cdot P_0 + d$

for equation (5d) in model 1.

Model 1C

for rule (5e) in model 1.

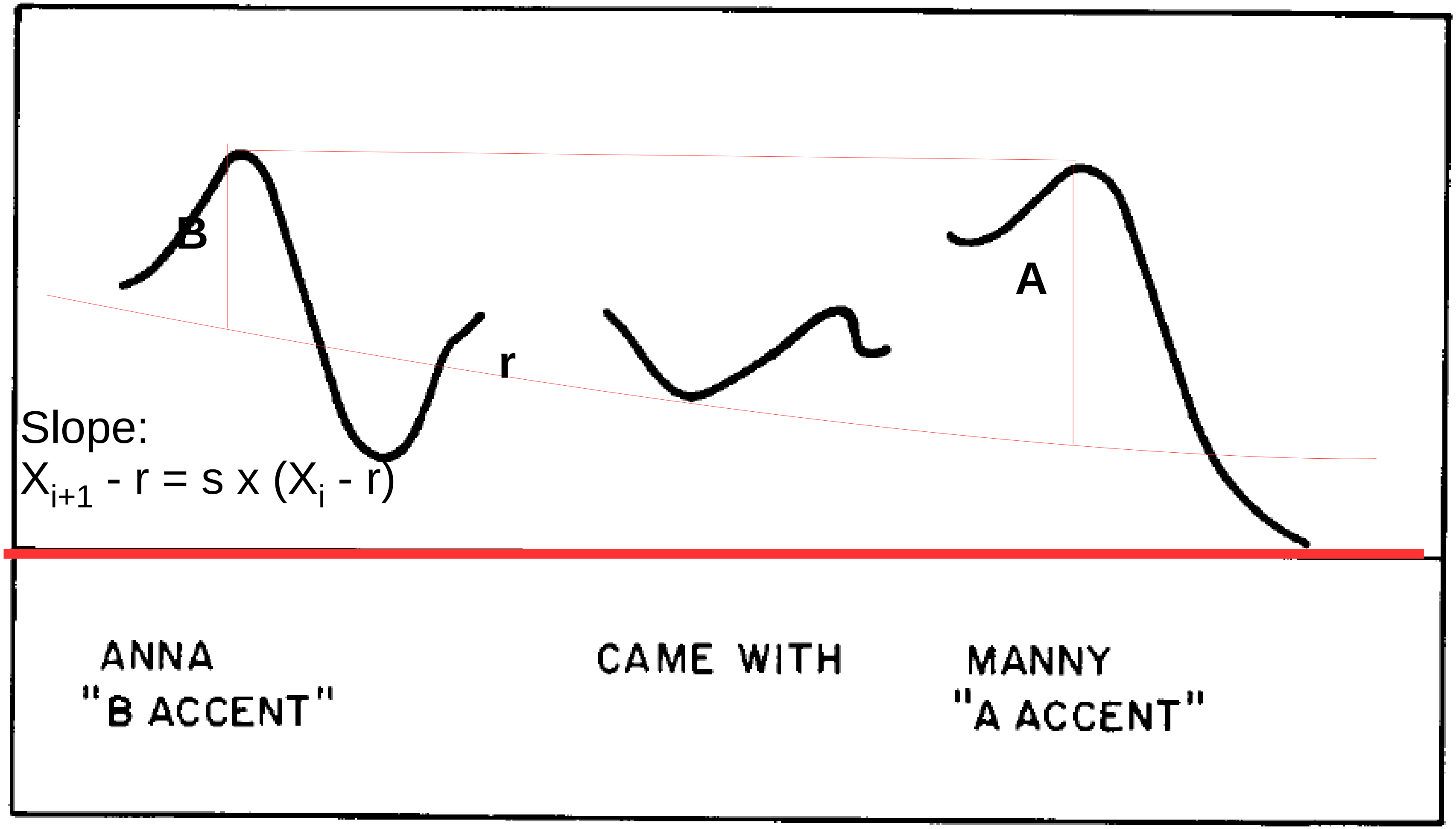
Models of f0 patterning: Liberman & Pierrehumbert



Slope:
$X_{i+1} - r = s \times (X_i - r)$

Figure 9
An F0 contour for *Anna came with Manny*, produced as a response to *What about Manny? Who came with him?*

Models of f0 patterning: Liberman & Pierrehumbert



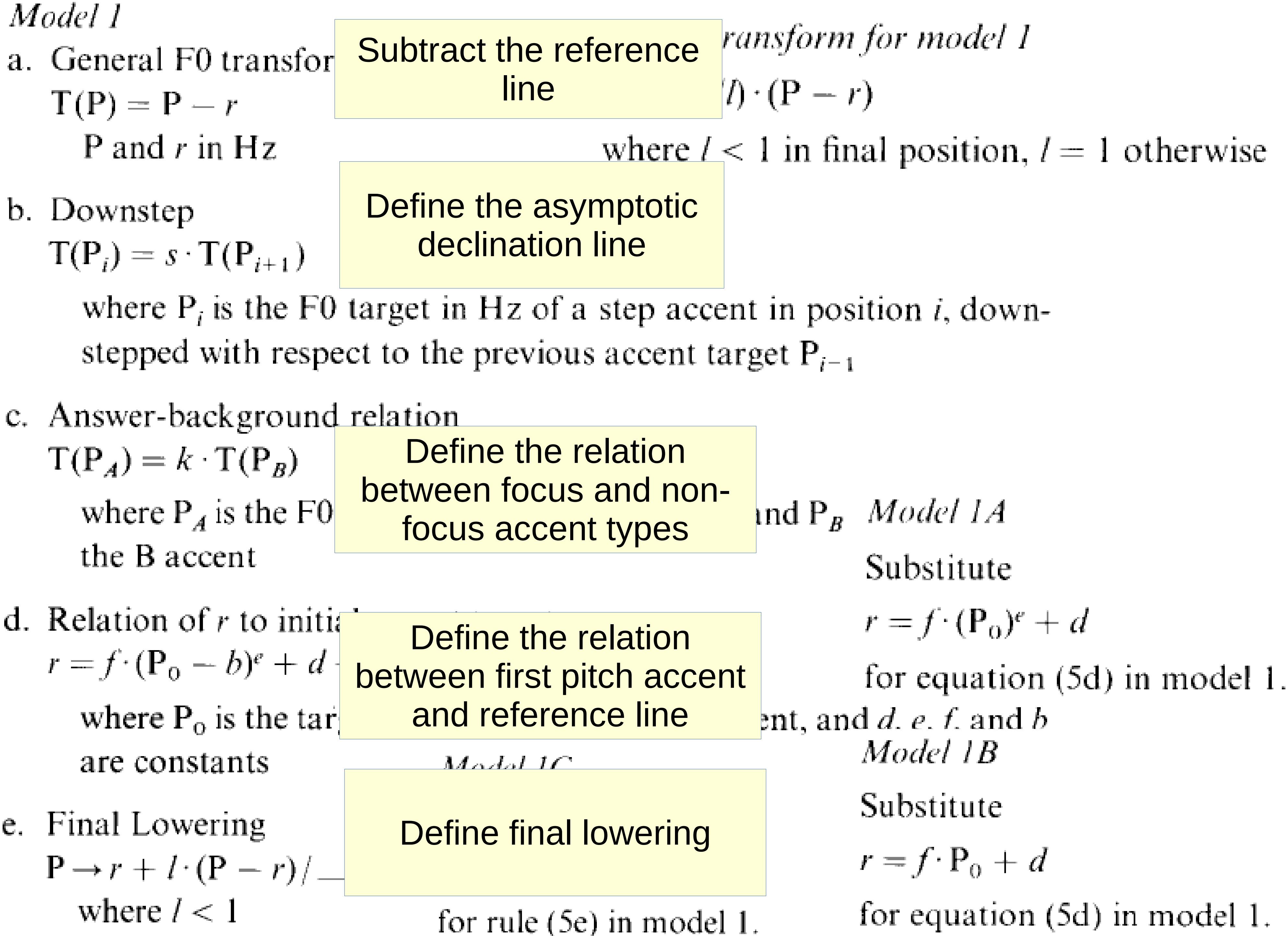
Slope:

$X_{i+1} - r = s \text{ x } (X_i - r)$

Figure 10

An F0 contour for *Anna came with Manny*, produced as a response to *What about Anna? Who did she come with?*

Models of f0 patterning: Liberman & Pierrehumbert



Model 1

a. General F0 transfor

$T(P) = P - r$

P and r in Hz

ransform for model 1

$l) \cdot (P - r)$

where $l < 1$ in final position, $l = 1$ otherwise

Subtract the reference line

b. Downstep

$T(P_i) = s \cdot T(P_{i+1})$

where P_i is the F0 target in Hz of a step accent in position i, downstepped with respect to the previous accent target P_{i-1}

Define the asymptotic declination line

c. Answer-background relation

$T(P_A) = k \cdot T(P_B)$

where P_A is the F0 and P_B the B accent

Define the relation between focus and non-focus accent types

Model 1A

Substitute

$r = f \cdot (P_0)^e + d$

for equation (5d) in model 1.

d. Relation of r to initi

$r = f \cdot (P_0 - b)^e + d$

where P_0 is the tar ent, and d, e, f, and b are constants

Define the relation between first pitch accent and reference line

Model 1B

Substitute

$r = f \cdot P_0 + d$

for equation (5d) in model 1.

Model 1C

e. Final Lowering

$P \rightarrow r + l \cdot (P - r) / \_\_$

where $l < 1$

for rule (5e) in model 1.

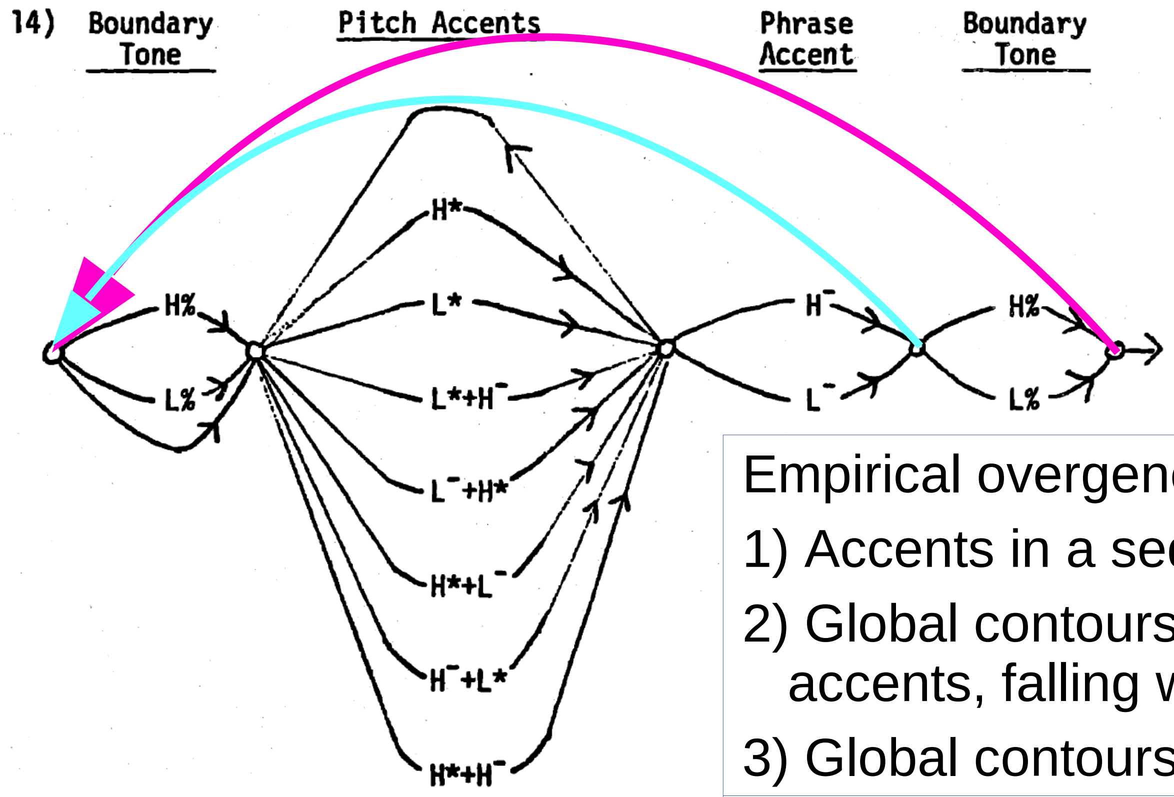
Define final lowering

Finite State Intonation Models

Intonational iteration as implementation of a layered hierarchy by means of of loops (linear abstract oscillations)

Pierrehumbert's regular grammar / finite state transition network

14)



Not the first (cf. Reich, 't Hart et al., Fujisaki, …)

But linguistically the most interesting.

Empirical overgeneration:

1) Accents in a sequence tend to be all H* or all L*

2) Global contours tend to be rising with L* accents, falling with H* accents

3) Global contours may span more than 1 turn

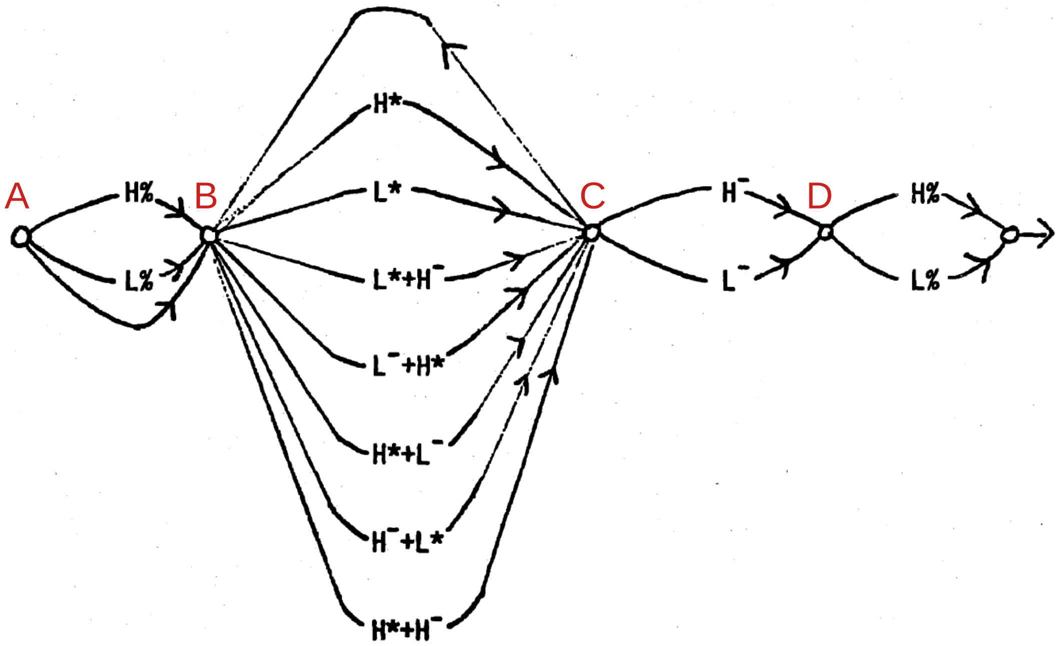
Empirical undergeneration:

1)Paratone hierarchy not included

2)No time constraints

Finite State Intonation Models

Intonation FST
(Pierrehumbert 1980)

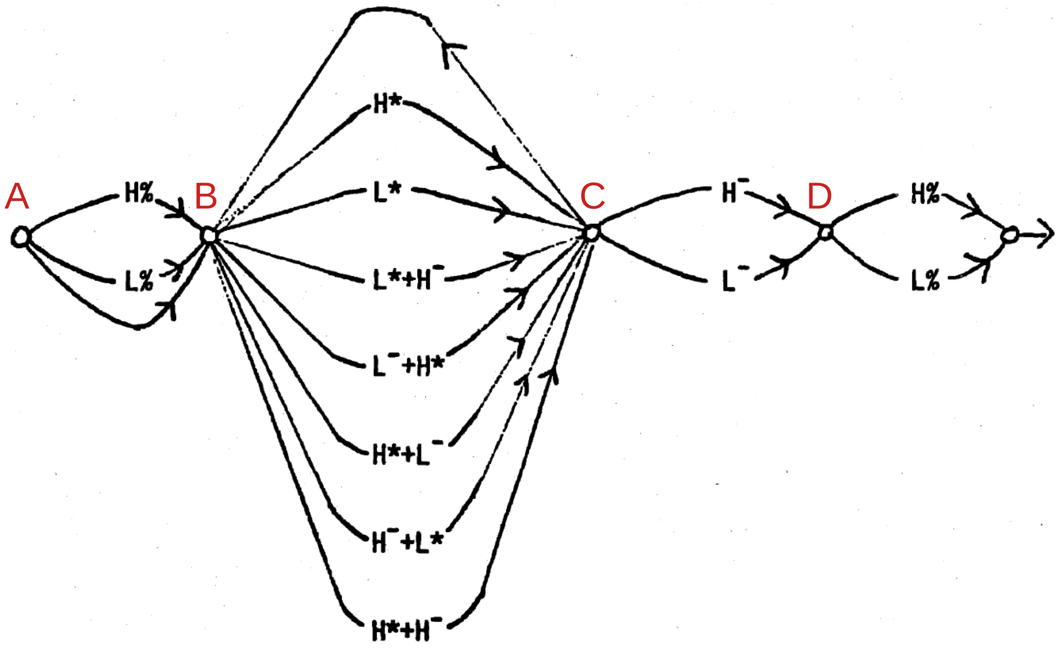


Equivalent right-branching Type 3 grammar:

A → ({ H%, L% }) B
B → { H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } { B, C}
C → { H-, L- } D
D → { H%, L% }

Finite State Intonation Models

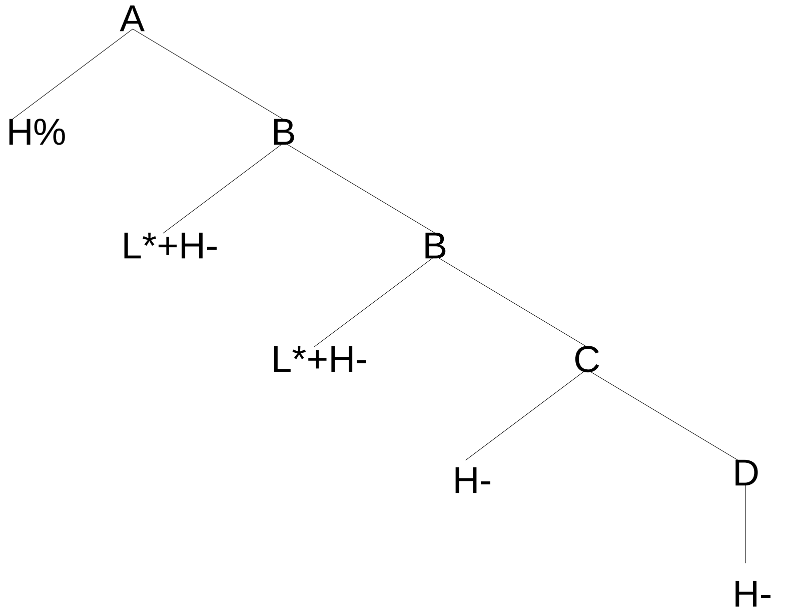
Intonation FST
(Pierrehumbert 1980)



Equivalent right-branching Type 3 grammar:

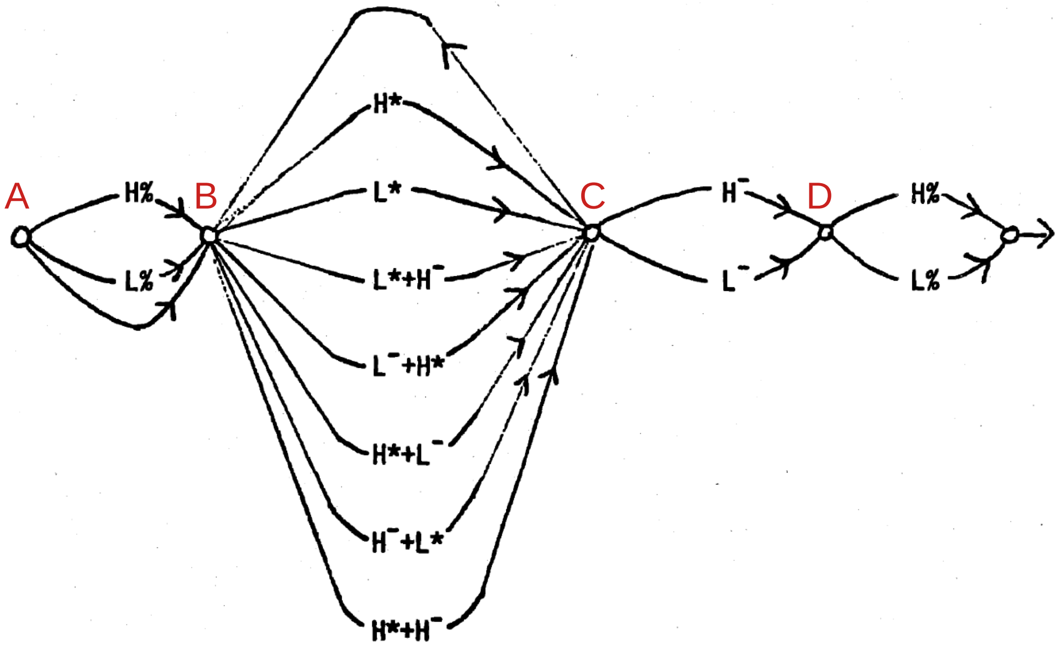
A → ({ H%, L% }) B
B → { H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } { B, C}
C → { H-, L- } D
D → { H%, L% }

Example of a right-branching tree based on this grammar:


A
H%
B
L*+H-
B
L*+H-
C
H-
D
H-

Finite State Intonation Models

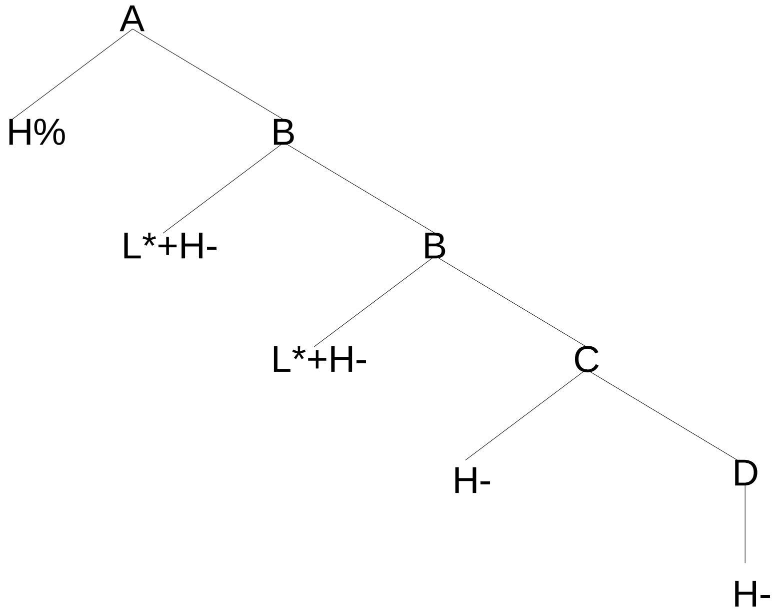
Intonation FST
(Pierrehumbert 1980)



Equivalent right-branching Type 3 grammar:

A → ({ H%, L% }) B
B → { H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } { B, C}
C → { H-, L- } D
D → { H%, L% }

Example of a right-branching tree based on this grammar:


A
H%
B
L*+H-
B
L*+H-
C
H-
D
H-

Equivalent regular expression:

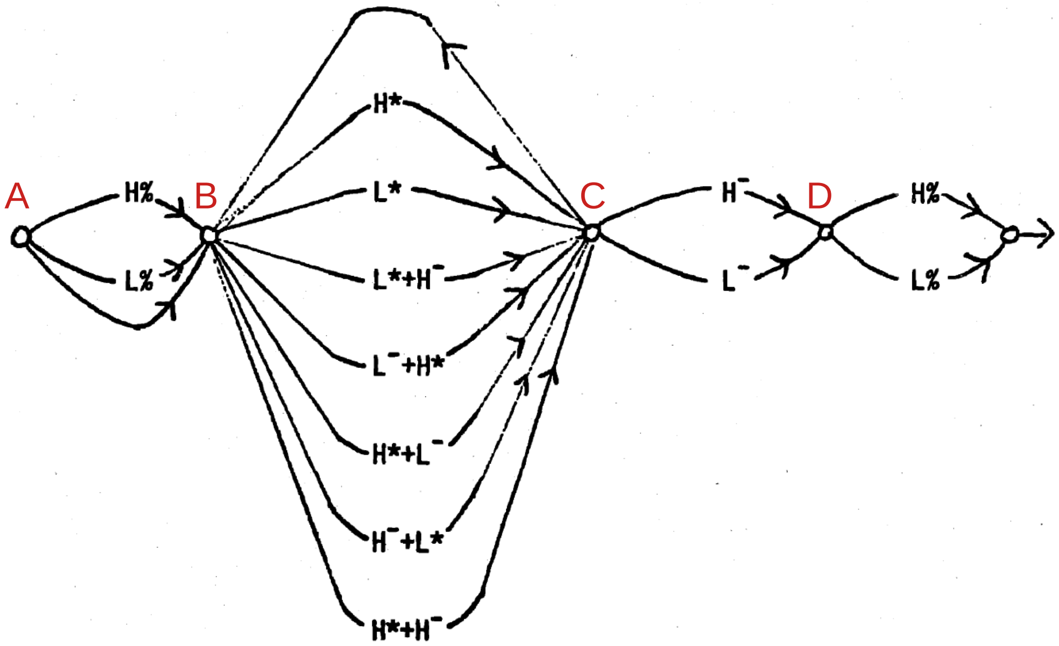
(H%|L%|ε) (H*|L*|L*H- |L-+H*|H*+L-|H-+L*| H*+H*|ε) (H-|L-) (H%|L%)

Abbreviated:

(BoundaryTone | ε) (PitchAccent | ε)* ipTone IPTone

Finite State Intonation Models

Intonation FST
(Pierrehumbert 1980)



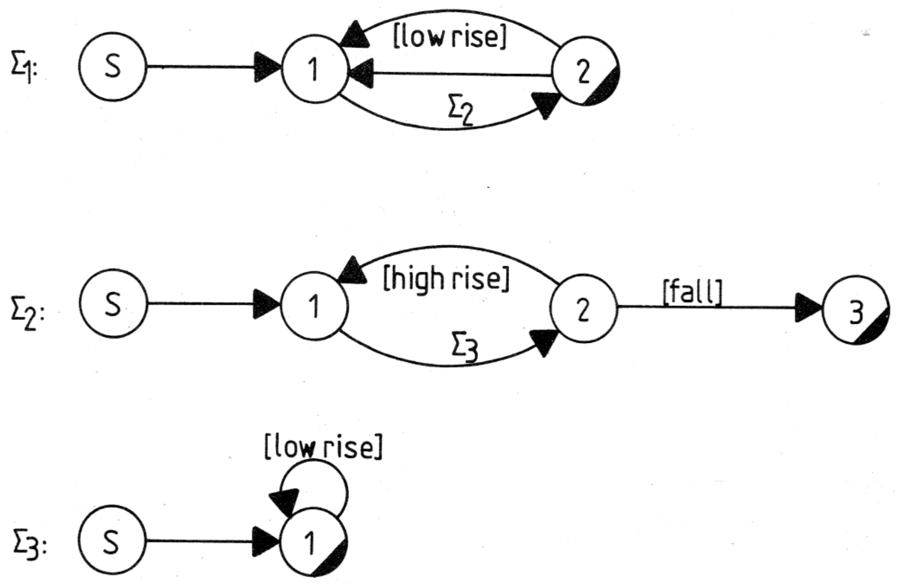
Equivalent regular expression:

(H%|L%|ε) (H*|L*|L*H- |L-+H*|H*+L-|H-+L*| H*+H*|ε) (H-|L-) (H%|L%)

Abbreviated:

(BoundaryTone | ε) (PitchAccent | ε)* ipTone IPTone

Subset of a Modular FST for the Intonation Hierarchy (Gibbon 1984)



Composed into an equivalent regular expression:

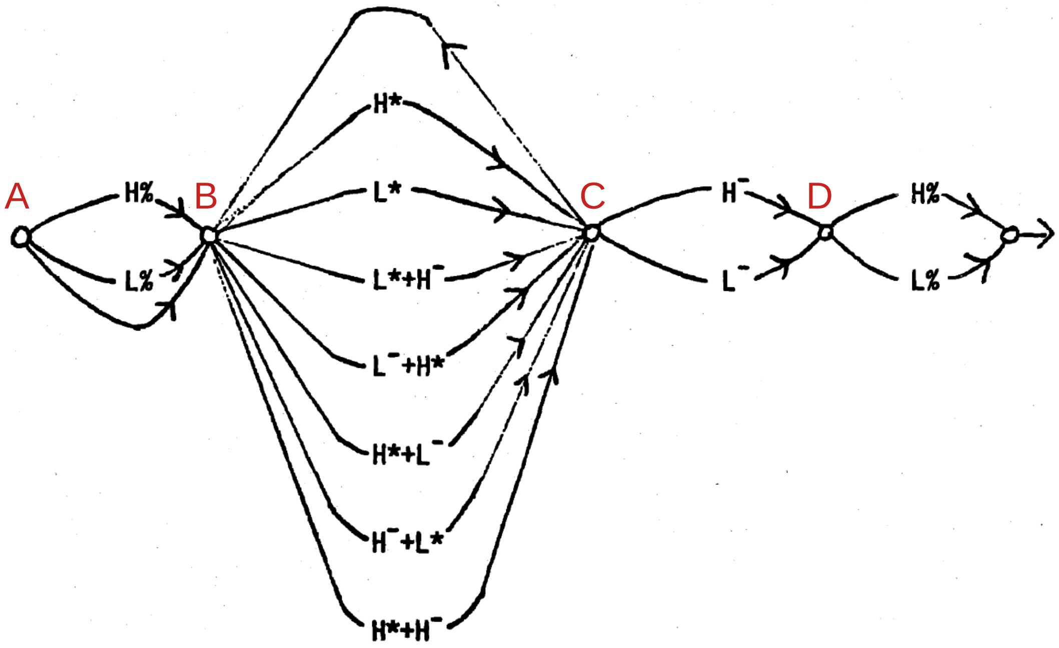
(([low_rise] * [high rise])* [fall] [low rise])*

Generalised:

((PitchAccent* PitchAccent) Nucleus Tail)*

A more general Finite State model

Intonation FST
(Pierrehumbert 1980)



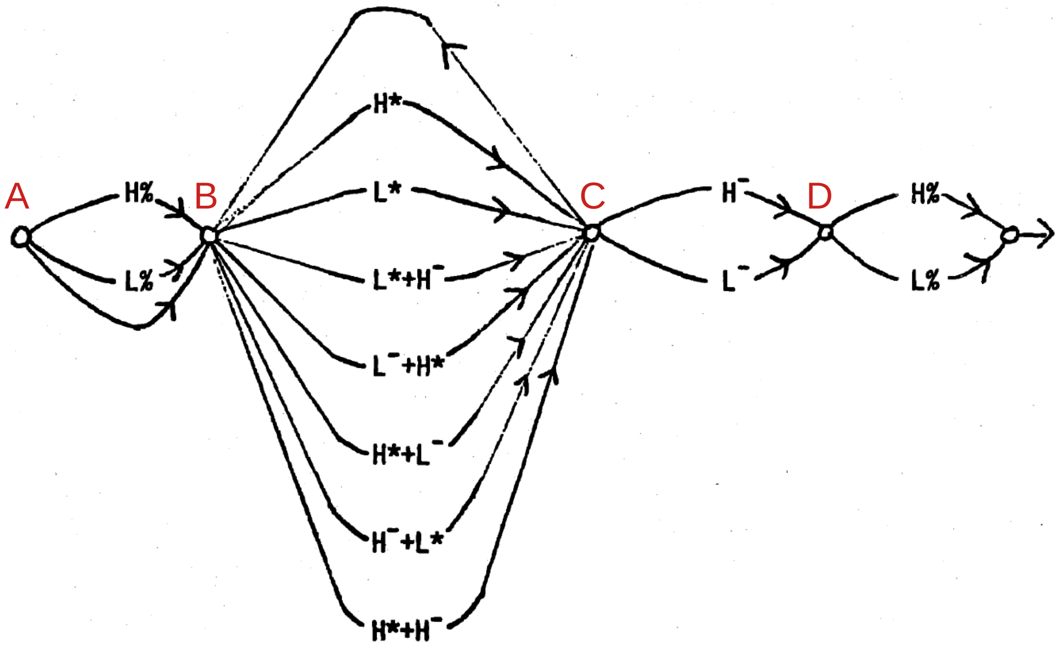
Generalisations:

1. Introduce *functional labels* into the grammar (cf. ‘Subject’, or ‘Nominative’ in sentences) to account for different contexts, e.g. ‘declination in declination’, taking Metalocutionary Theory into account
2. Create a sublexicon for each pitch accent and boundary tone type
3. Add functional label options to each pitch accent and tone type in each sublexicon
4. Create a lexicon out of the union of sublexica*

(the version below omits the functional labels)

A more general Finite State model

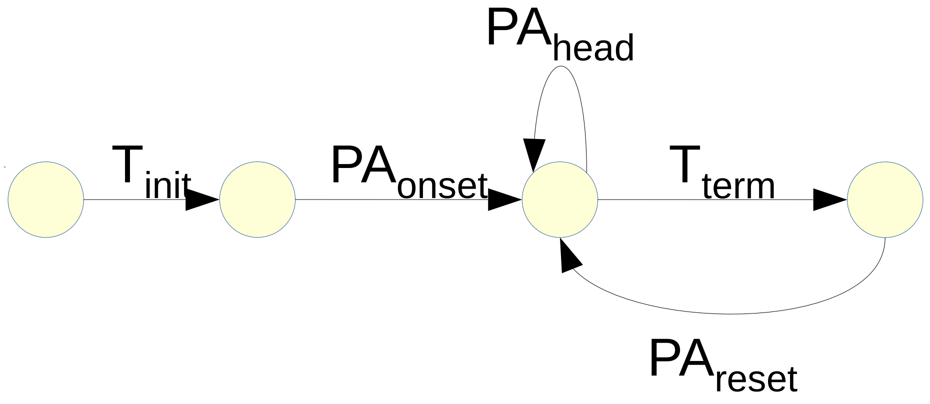
Intonation FST
(Pierrehumbert 1980)



Generalisations:

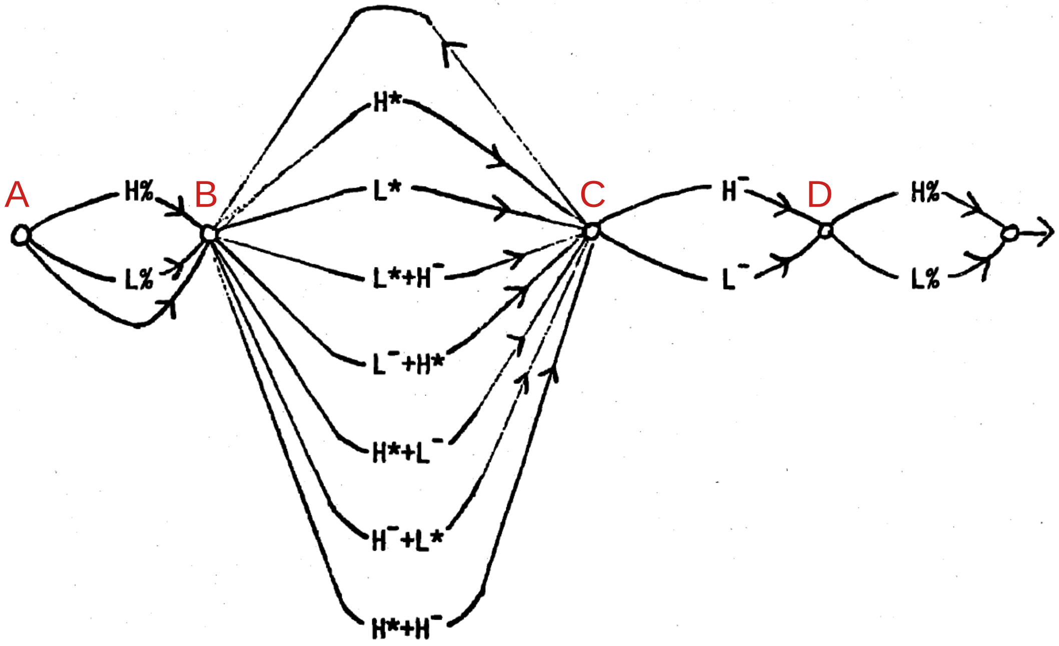
1. Introduce *functional labels* into the grammar (cf. ‘Subject’, or ‘Nominative’ in sentences) to account for different contexts, e.g. ‘declination in declination’, taking Metalocutionary Theory into account
2. Create a sublexicon for each pitch accent and boundary tone type
3. Add functional label options to each pitch accent and tone type in each sublexicon
4. Create a lexicon out of the union of sublexica*

(the version below omits the functional labels)



A more general Finite State model

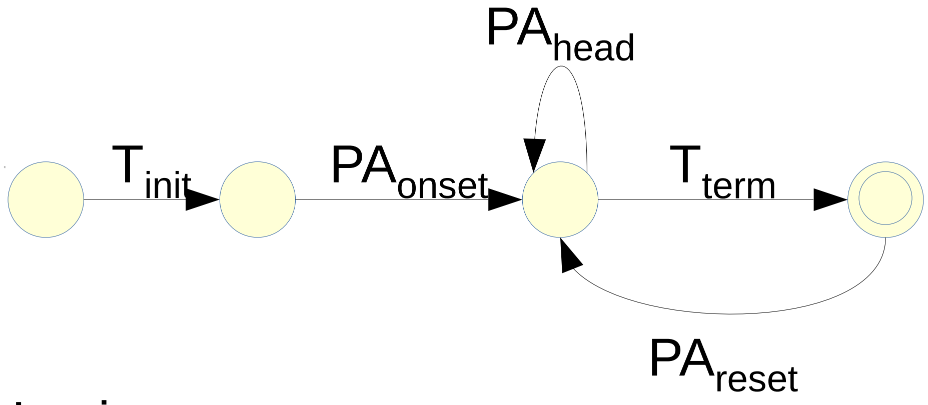
Intonation FST
(Pierrehumbert 1980)



Generalisations:

1. Introduce *functional labels* into the grammar (cf. ‘Subject’, or ‘Nominative’ in sentences) to account for different contexts, e.g. ‘declination in declination’, taking Metalocutionary Theory into account
2. Create a sublexicon for each pitch accent and boundary tone type
3. Add functional label options to each pitch accent and tone type in each sublexicon
4. Create a lexicon out of the union of sublexica*

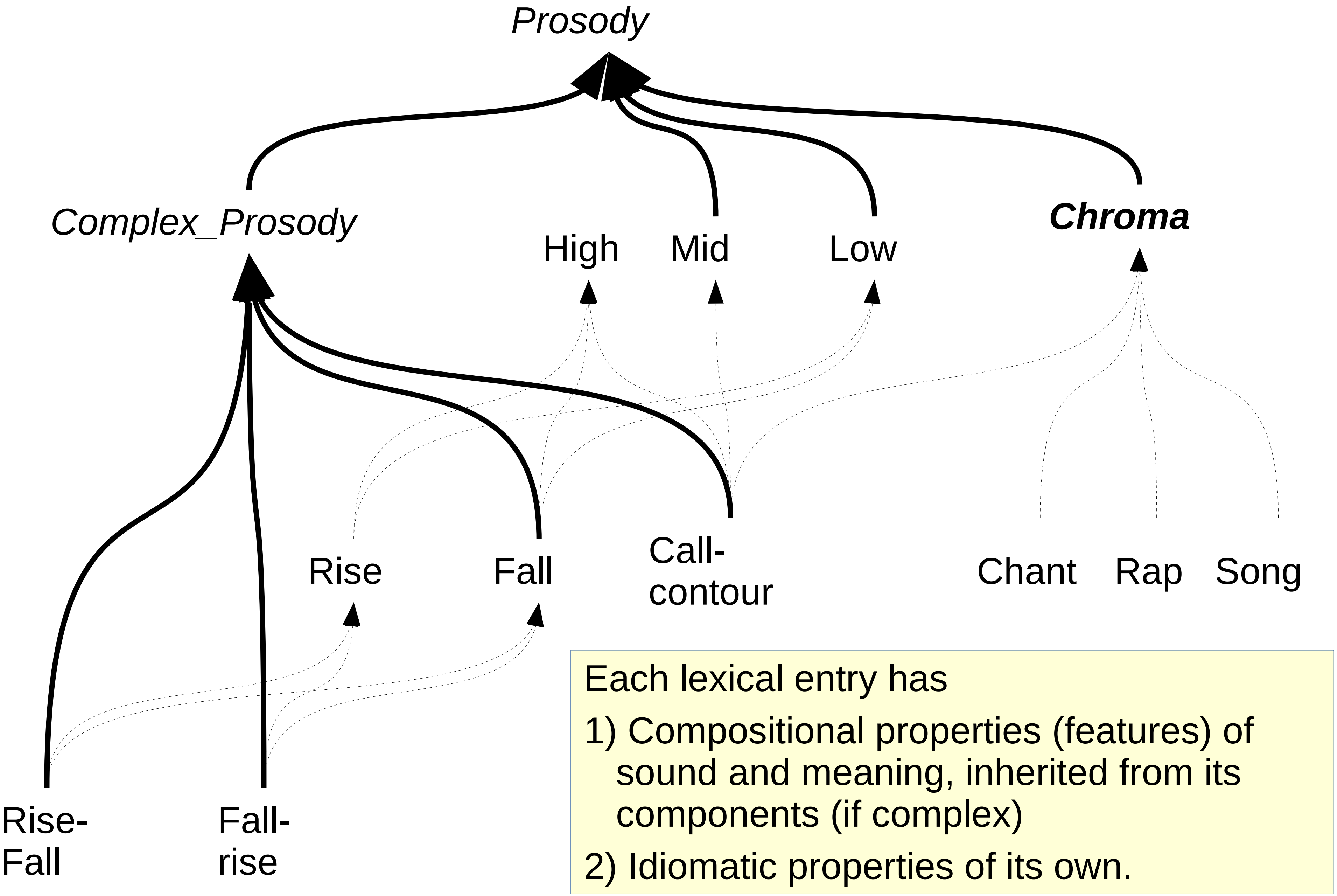
   (the version below omits the functional labels)



Lexicon$_{PA}$ =

{ H%, L% } ∪
{ H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } ∪
{ H-, L-} ∪
{ H%, L% }

Default inheritance lexicon for English pitch accents



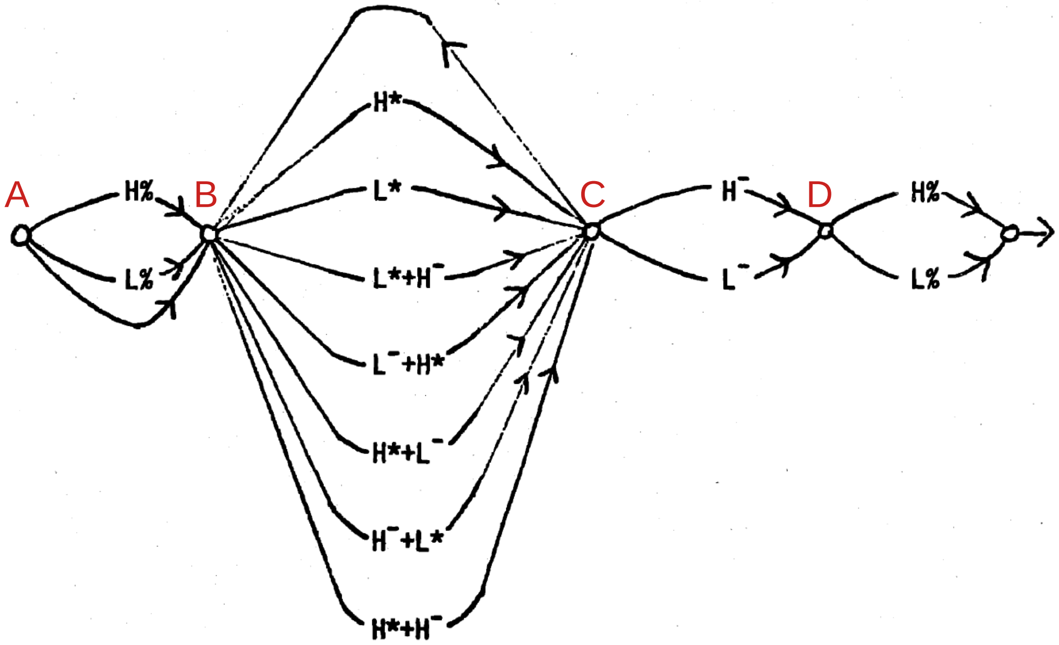
Each lexical entry has

1) Compositional properties (features) of sound and meaning, inherited from its components (if complex)

2) Idiomatic properties of its own.

A more general Finite State model

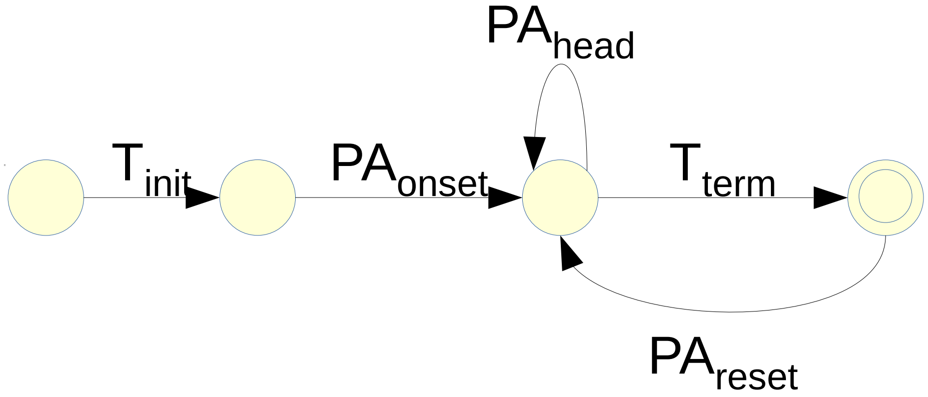
Intonation FST
(Pierrehumbert 1980)



Generalisations:

1. Introduce *functional labels* into the grammar (cf. ‘Subject’, or ‘Nominative’ in sentences) to account for different contexts, e.g. ‘declination in declination’, taking Metalocutionary Theory into account
2. Create a sublexicon for each pitch accent and boundary tone type
3. Add functional label options to each pitch accent and tone type in each sublexicon
4. Create a lexicon out of the union of sublexica*

(the version below omits the functional labels)



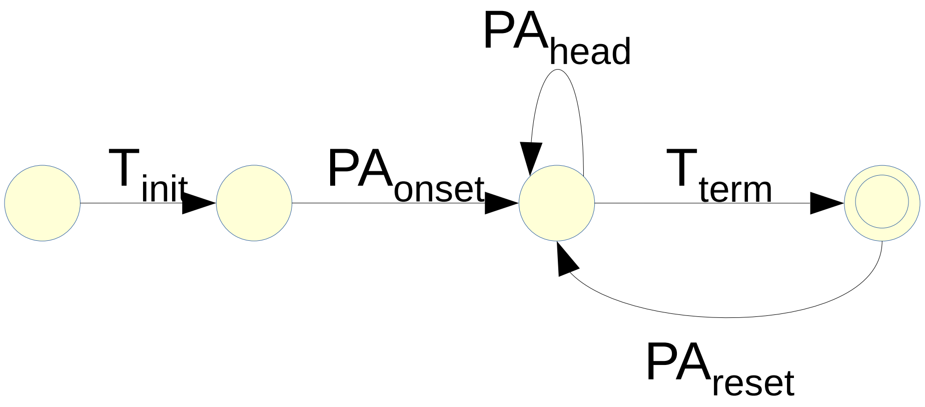
Modality interpretation variables at each node
Locutionary constraints at all nodes:

- logical operators (if-then, and, but, ...)
- information structure
- dialogue patterns

Lexicon_{PA} =

{ H%, L% } ∪
{ H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } ∪
{ H-, L-} ∪
{ H%, L% }

A more general Finite State model


$\mathrm{PA}_{\mathrm{head}}$
$\mathrm{T}_{\mathrm{init}}$
$\mathrm{PA}_{\mathrm{onset}}$
$\mathrm{T}_{\mathrm{term}}$
$\mathrm{PA}_{\mathrm{reset}}$

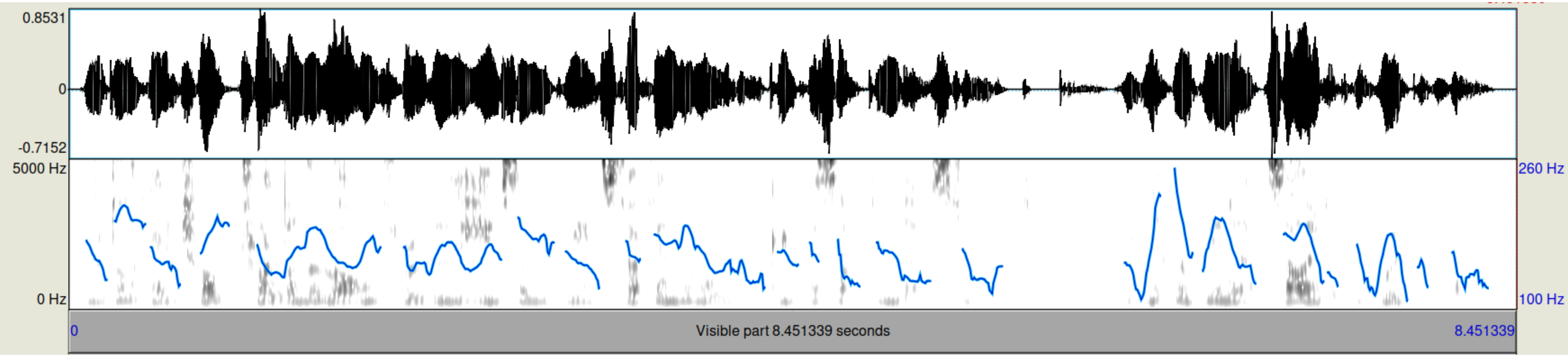
Lexicon$_{\mathrm{PA}}$ =

{ H%, L% } ∪
{ H*, L*, L*H-, L-+H*, H*+L-, H-+L*, H*+H* } ∪
{ H-, L-} ∪
{ H%, L% }

Generalisations:

1. Introduce *functional labels* into the grammar (cf. ‘Subject’, or ‘Nominative’ in sentences) to account for different contexts, e.g. ‘declination in declination’, taking Metalocutionary Theory into account
2. Create a sublexicon for each pitch accent and boundary tone type
3. Add functional label options to each pitch accent and tone type in each sublexicon
4. Create a lexicon out of the union of sublexica*

(the version below omits the functional labels)

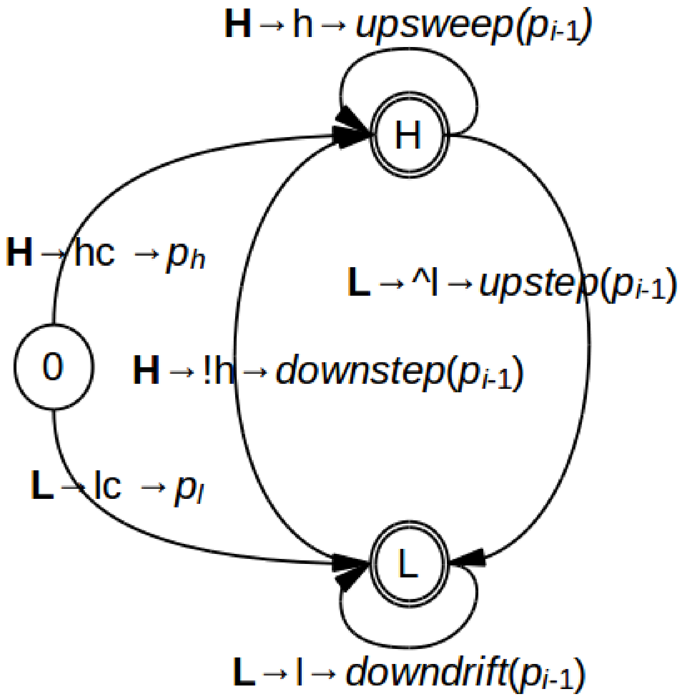


Little Hedgehog trundled along through the leaves and the green stuff in the wood, looking for something nice to eat.

He’d never been outside of the wood before.

Finite State Tone Models of Tone Sandhi

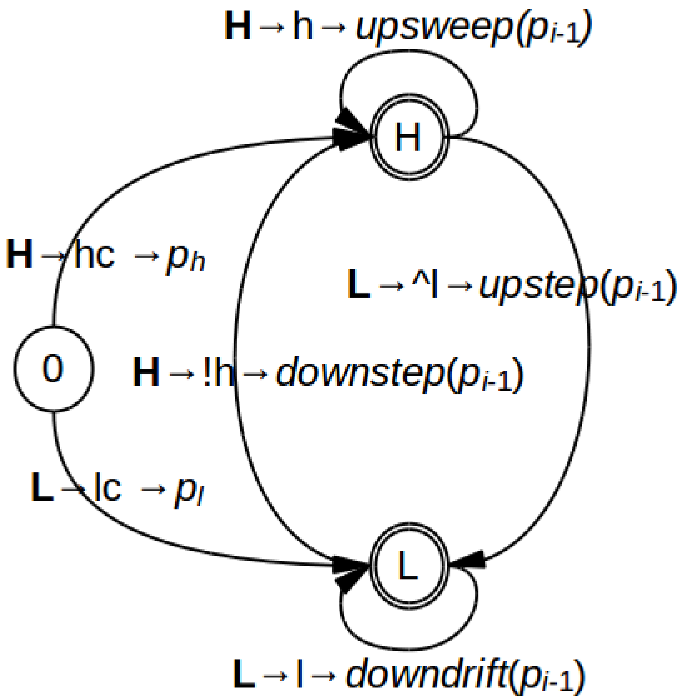
Tone Sandhi FST for a 2-tone Niger-Congo language
(Gibbon 1987, 2001)

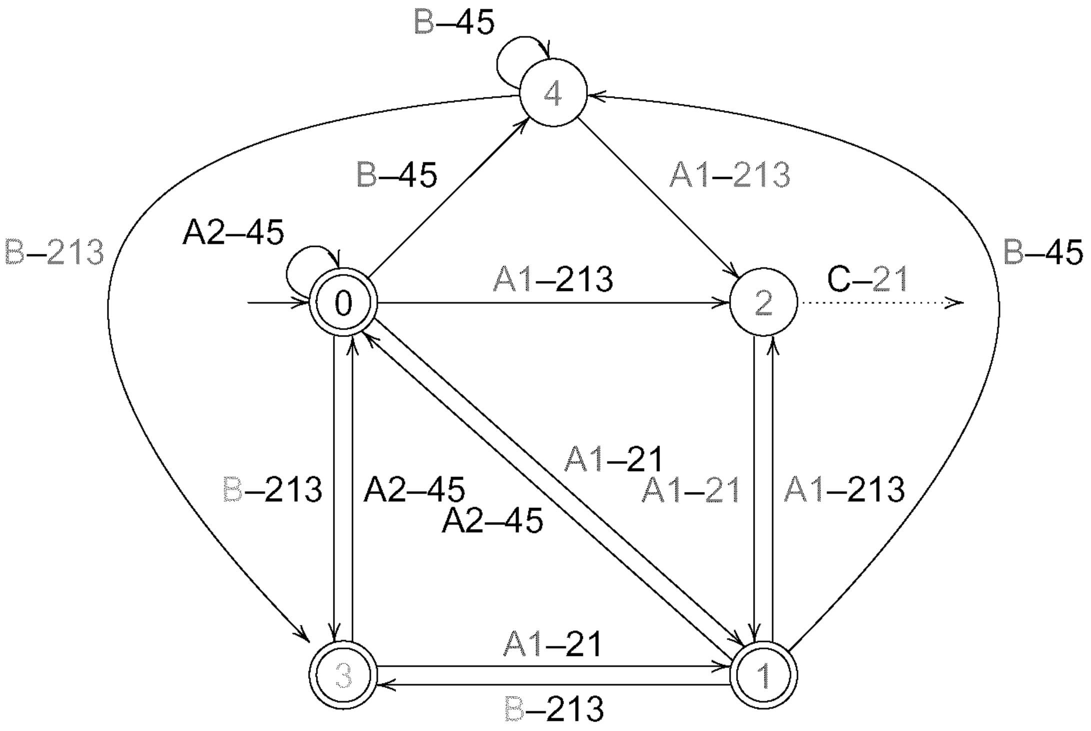

H → h → upsweep(p_{i-1})
H
H → hc → p_h
L → ^l → upstep(p_{i-1})
0
H → !h → downstep(p_{i-1})
L → lc → p_l
L
L → l → downdrift(p_{i-1})

Finite State Tone Models of Tone Sandhi

Tone Sandhi FST for a 2-tone Niger-Congo language
(Gibbon 1987, 2001)

Tone Sandhi FST for Tianjin Mandarin
(Jansche 1998)


H → h → *upsweep*(p_{i-1})
H
H → hc → p_h
L → ^l → *upstep*(p_{i-1})
0
H → !h → *downstep*(p_{i-1})
L → lc → p_l
L
L → l → *downdrift*(p_{i-1})



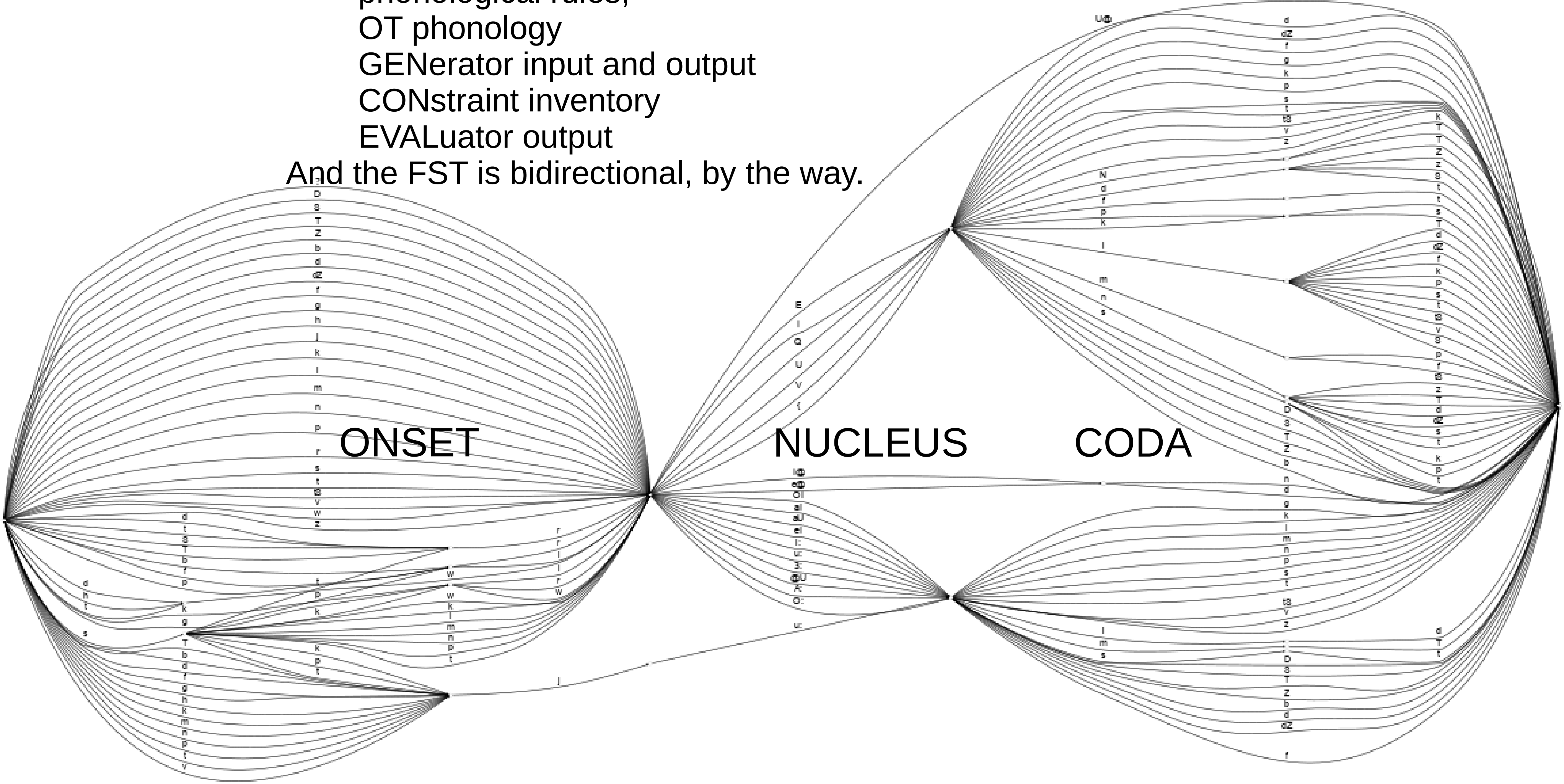
How to bring rules and constraints together

This Finite State Network for English Syllables defines all syllable-internal contexts.

For example for ...

- contextual variation
- tone assignment
- phonological rules,
- OT phonology
- GENerator input and output
- CONstraint inventory
- EVALuator output

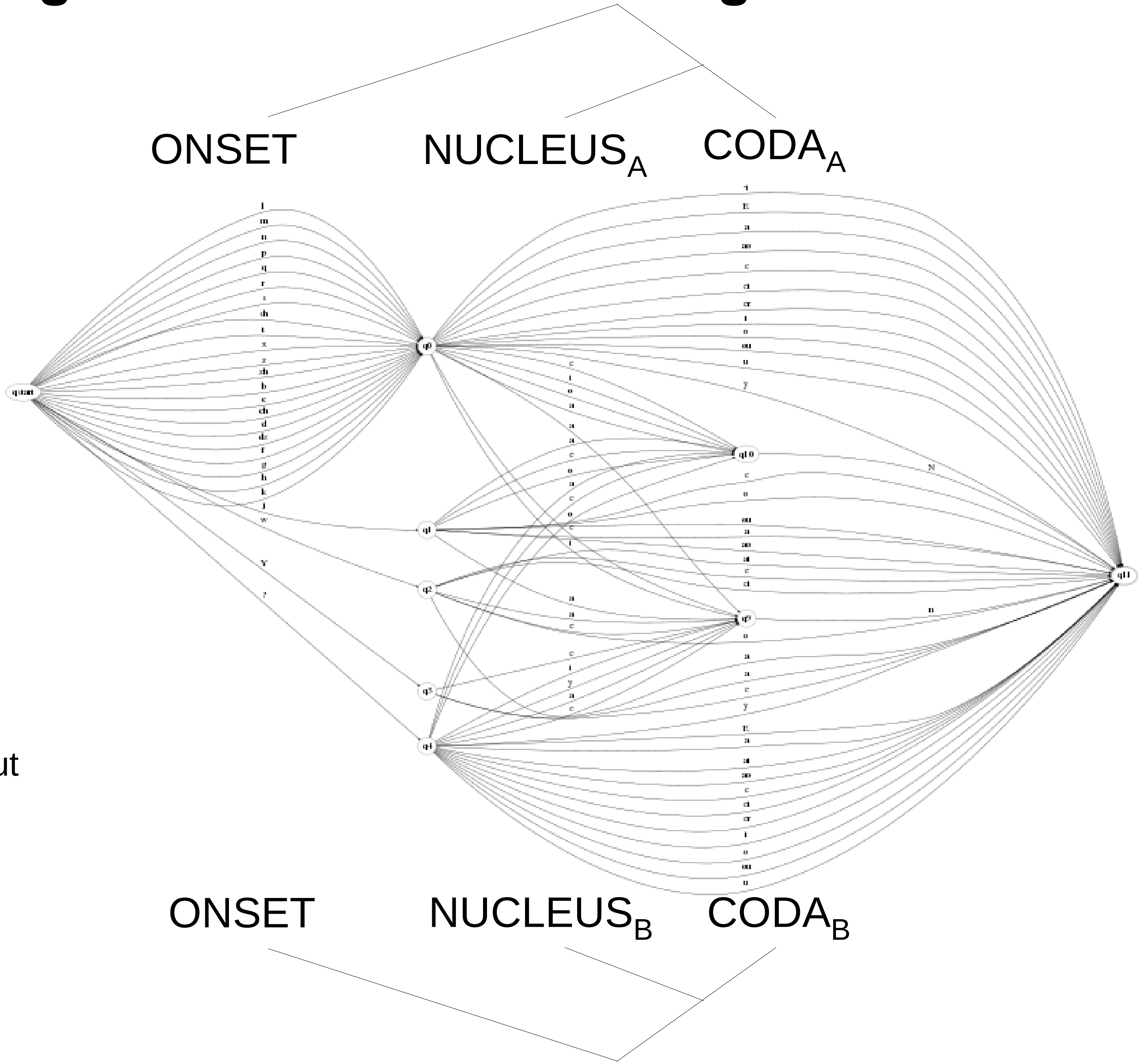
And the FST is bidirectional, by the way.



http://wwwhomes.uni-bielefeld.de/gibbon/Syllables/english-syllables-demo.html

How to bring rules and constraints together

This Finite State Network for Chinese syllables defines all syllable-internal contexts.
Tones are not shown.



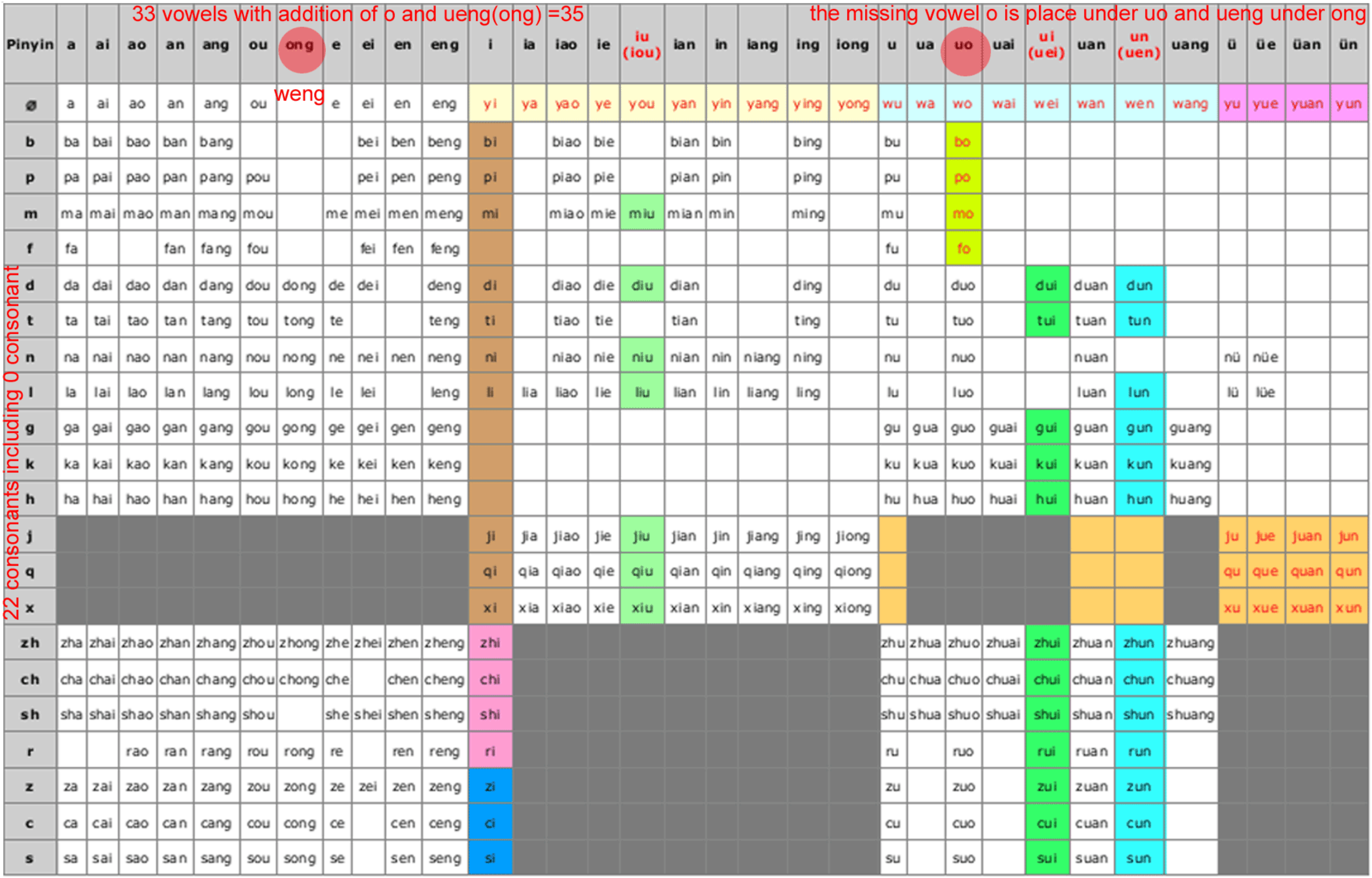
For example for ...

- contextual variation
- tone assignment
- phonological rules,
- OT phonology
- GENerator input and output
- CONstraint inventory
- EVALuator output

And the FST is bidirectional, by the way.

https://wwwhomes.uni-bielefeld.de/gibbon/Syllables/Mandarin/

Pinyin initials-finals table



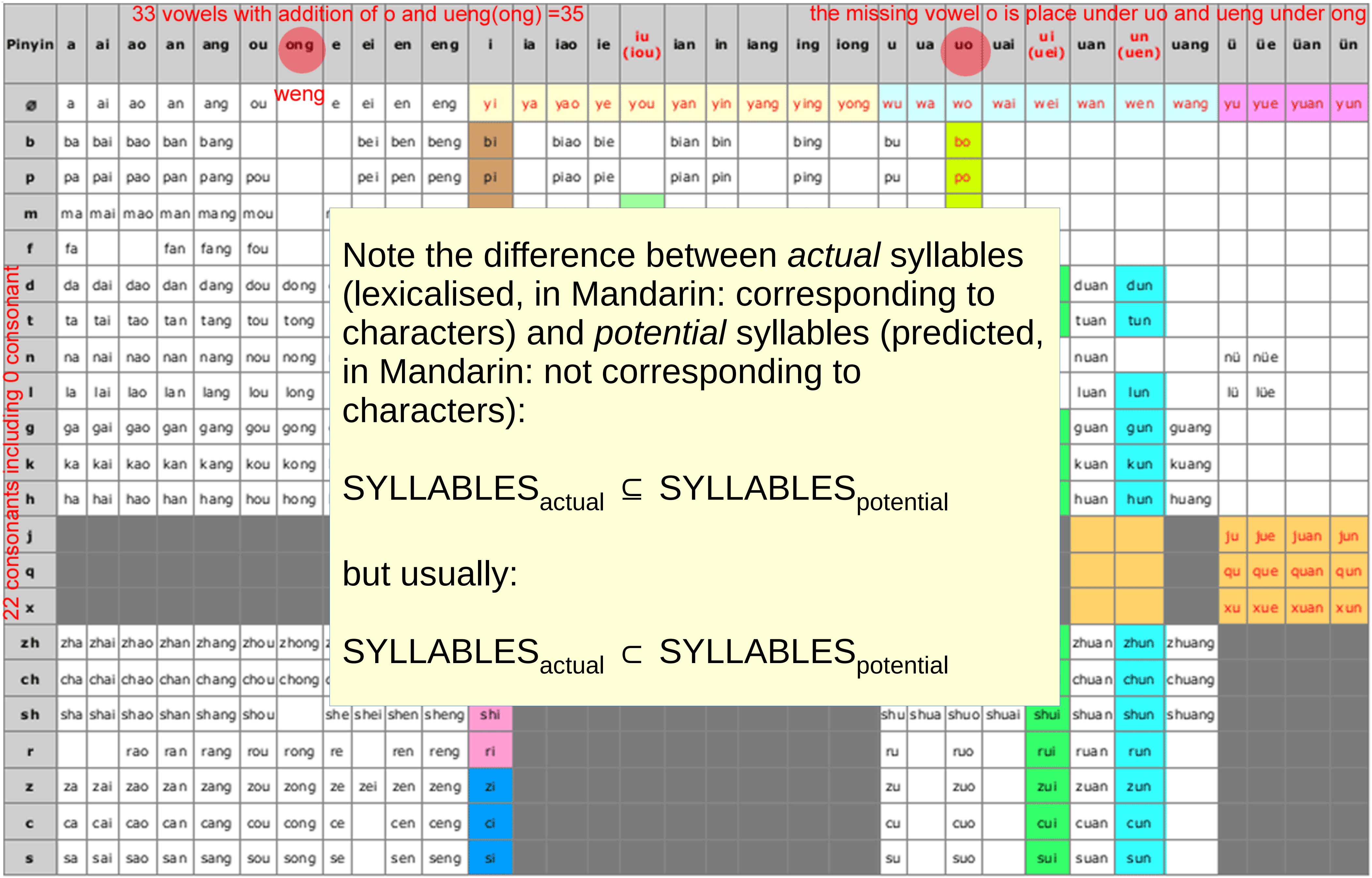
33 vowels with addition of o and ueng(ong) =35

the missing vowel o is place under uo and ueng under ong

22 consonants including 0 consonant

Pinyin	a	ai	ao	an	ang	ou	ong	e	ei	en	eng	i	ia	iao	ie	iu (iou)	ian	in	iang	ing	iong	u	ua	uo	uai	ui (uei)	uan	un (uen)	uang	ü	üe	üan	ün
ø	a	ai	ao	an	ang	ou	weng	e	ei	en	eng	yi	ya	yao	ye	you	yan	yin	yang	ying	yong	wu	wa	wo	wai	wei	wan	wen	wang	yu	yue	yuan	yun
b	ba	bai	bao	ban	bang				bei	ben	beng	bi		biao	bie		bian	bin		bing		bu		bo									
p	pa	pai	pao	pan	pang	pou			pei	pen	peng	pi		piao	pie		pian	pin		ping		pu		po									
m	ma	mai	mao	man	mang	mou		me	mei	men	meng	mi		miao	mie	miu	mian	min		ming		mu		mo									
f	fa			fan	fang	fou			fei	fen	feng											fu		fo									
d	da	dai	dao	dan	dang	dou	dong	de	dei		deng	di		diao	die	diu	dian			ding		du		duo		dui	duan	dun					
t	ta	tai	tao	tan	tang	tou	tong	te			teng	ti		tiao	tie		tian			ting		tu		tuo		tui	tuan	tun					
n	na	nai	nao	nan	nang	nou	nong	ne	nei	nen	neng	ni		niao	nie	niu	nian	nin	niang	ning		nu		nuo			nuan			nü	nüe		
l	la	lai	lao	lan	lang	lou	long	le	lei		leng	li	lia	liao	lie	liu	lian	lin	liang	ling		lu		luo			luan	lun		lü	lüe		
g	ga	gai	gao	gan	gang	gou	gong	ge	gei	gen	geng											gu	gua	guo	guai	gui	guan	gun	guang				
k	ka	kai	kao	kan	kang	kou	kong	ke	kei	ken	keng											ku	kua	kuo	kuai	kui	kuan	kun	kuang				
h	ha	hai	hao	han	hang	hou	hong	he	hei	hen	heng											hu	hua	huo	huai	hui	huan	hun	huang				
j												ji	jia	jiao	jie	jiu	jian	jin	jiang	jing	jiong									ju	jue	juan	jun
q												qi	qia	qiao	qie	qiu	qian	qin	qiang	qing	qiong									qu	que	quan	qun
x												xi	xia	xiao	xie	xiu	xian	xin	xiang	xing	xiong									xu	xue	xuan	xun
zh	zha	zhai	zhao	zhan	zhang	zhou	zhong	zhe	zhei	zhen	zheng	zhi										zhu	zhua	zhuo	zhuai	zhui	zhuan	zhun	zhuang				
ch	cha	chai	chao	chan	chang	chou	chong	che		chen	cheng	chi										chu	chua	chuo	chuai	chui	chuan	chun	chuang				
sh	sha	shai	shao	shan	shang	shou		she	shei	shen	sheng	shi										shu	shua	shuo	shuai	shui	shuan	shun	shuang				
r			rao	ran	rang	rou	rong	re		ren	reng	ri										ru		ruo		rui	ruan	run					
z	za	zai	zao	zan	zang	zou	zong	ze	zei	zen	zeng	zi										zu		zuo		zui	zuan	zun					
c	ca	cai	cao	can	cang	cou	cong	ce		cen	ceng	ci										cu		cuo		cui	cuan	cun					
s	sa	sai	sao	san	sang	sou	song	se		sen	seng	si										su		suo		sui	suan	sun					

Pinyin initials-finals table



33 vowels with addition of o and ueng(ong) =35

the missing vowel o is place under uo and ueng under ong

22 consonants including 0 consonant

Pinyin	a	ai	ao	an	ang	ou	ong	e	ei	en	eng	i	ia	iao	ie	iu (iou)	ian	in	iang	ing	iong	u	ua	uo	uai	ui (uei)	uan	un (uen)	uang	ü	üe	üan	ün
ø	a	ai	ao	an	ang	ou	weng	e	ei	en	eng	yi	ya	yao	ye	you	yan	yin	yang	ying	yong	wu	wa	wo	wai	wei	wan	wen	wang	yu	yue	yuan	yun
b	ba	bai	bao	ban	bang				bei	ben	beng	bi		biao	bie		bian	bin		bing		bu		bo									
p	pa	pai	pao	pan	pang	pou			pei	pen	peng	pi		piao	pie		pian	pin		ping		pu		po									
m	ma	mai	mao	man	mang	mou																											
f	fa			fan	fang	fou																											
d	da	dai	dao	dan	dang	dou	dong																				duan	dun					
t	ta	tai	tao	tan	tang	tou	tong																				tuan	tun					
n	na	nai	nao	nan	nang	nou	nong																				nuan			nü	nüe		
l	la	lai	lao	lan	lang	lou	long																				luan	lun		lü	lüe		
g	ga	gai	gao	gan	gang	gou	gong																				guan	gun	guang				
k	ka	kai	kao	kan	kang	kou	kong																				kuan	kun	kuang				
h	ha	hai	hao	han	hang	hou	hong																				huan	hun	huang				
j																														ju	jue	juan	jun
q																														qu	que	quan	qun
x																														xu	xue	xuan	xun
zh	zha	zhai	zhao	zhan	zhang	zhou	zhong																				zhuan	zhun	zhuang				
ch	cha	chai	chao	chan	chang	chou	chong																				chuan	chun	chuang				
sh	sha	shai	shao	shan	shang	shou		she	shei	shen	sheng	shi										shu	shua	shuo	shuai	shui	shuan	shun	shuang				
r			rao	ran	rang	rou	rong	re		ren	reng	ri										ru		ruo		rui	ruan	run					
z	za	zai	zao	zan	zang	zou	zong	ze	zei	zen	zeng	zi										zu		zuo		zui	zuan	zun					
c	ca	cai	cao	can	cang	cou	cong	ce		cen	ceng	ci										cu		cuo		cui	cuan	cun					
s	sa	sai	sao	san	sang	sou	song	se		sen	seng	si										su		suo		sui	suan	sun					

Note the difference between *actual* syllables (lexicalised, in Mandarin: corresponding to characters) and *potential* syllables (predicted, in Mandarin: not corresponding to characters):

$$\text{SYLLABLES}_{\text{actual}} \subseteq \text{SYLLABLES}_{\text{potential}}$$

but usually:

$$\text{SYLLABLES}_{\text{actual}} \subset \text{SYLLABLES}_{\text{potential}}$$

Some analyses in the literature have not quite got it ...

A widespread view

“This paper examines the hypothesis that higher prosodic constituents are recursive.”

Problems

The concept of recursion is either not clearly defined, or re-defined *ad hoc*.

But, as we saw, there are several kinds of recursion, each of which have different degrees of complexity, and have quite different implications for the processing of human speech.

Incidentally, it is very important in this context to distinguish between the processing of speech and the processing of writing. For the latter, additional external memory is available.

Many analyses are made on the basis of transcriptions – but these are writing!

Two definitions of recursion

Féry, Caroline. 2010. Recursion in prosodic structure. *Linguistics*.

Hauser, Chomsky & Fitch (2002) define recursion as the basic operation that allows the generation of a potentially infinite array of discrete expressions out of a finite set of elements. The set of finite elements is hierarchically organized.

For prosody, recursion implies a set of prosodic domains which can be repeated at each level of the hierarchy. We already saw that lower prosodic domains cannot dominate higher ones. Either the domains are repeated linearly, or they are contained within each other. The former method is known as iteration, and is universally admitted in the literature on prosodic structure. It is illustrated in (6) with a list, see for instance Nespor & Vogel (1986), Liberman & Pierrehumbert (1984), van Heuven (2004) for the prosodic realization of lists. In an iterative structure, as in (6), the prosodic domains iterate but do not overlap.

(6) (*Anna made some errands and bought*) [*a bottle of orange juice*] P, [an apple] P, [sugar] P, [butter] P, [a pair of socks] P

Comment

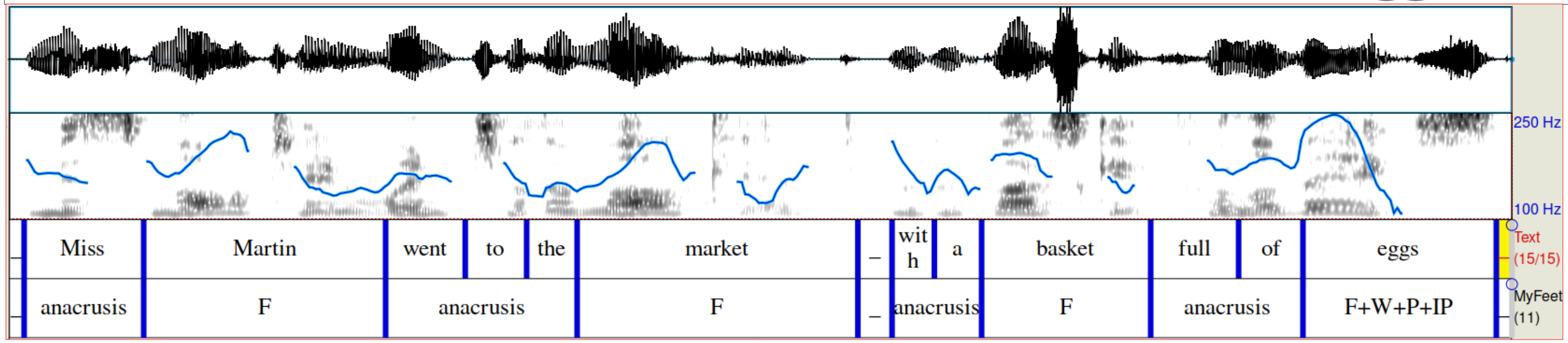
This means that the example can be described by a head or tail recursive Type 3 grammar, or an iterative finite state automaton.

The first case

Strict Layer Hypothesis

Here from Féry 2010, Recursion in Prosodic Structure. *Linguistics*.

```
7  (                                                          ×   )    IP
6  (     ×     ) (               ×   )   (      ×     ) (     ×   )    p-phrase
5  ( × )  (×    ) (×   ) (        ×     )   (      ×     ) (×    )( ×  )    PW
4    F     F       F    F    F            F    F      F     F
3    σ     σ   σ   σ   σ   σ σ   σ        σ   σ σ   σ   σ    σ  σ
2   μμ    μμ μμ   μμ μ   μ μμ   μμ      μμ μ μμ μμ μμ   μμ μμ
1  Miss   Martin went to the market      with a basket full   of eggs
```



In this example, there is a finite depth of 7 (including terminal elements), so Type 3?

The second case: centre-embedding?

Féry, Caroline. 2010. Recursion in prosodic structure. *Linguistics*.

But here, the second meaning of recursion will be investigated: a prosodic domain of level n may be contained in another, larger domain of the same type n. We thus make a principled distinction between iteration of prosodic domains n, and recursion of prosodic domains n (see also Hunyadi (2006) for this distinction). In recursion proper, a center-embedded clause occurs in the middle of a main clause, which, as a result, is divided into two parts.

Comment

Based on previous discussion, this claim implies that there are

1. prosodic structures which are centre-embedding and not linear or iterative,
2. prosodic domains which must be described by a Type 2 grammar (maybe even a Type 1 grammar or a Type 0 grammar) and cannot be described by a Type 3 grammar,
3. prosodic structures which require more than finite memory,
4. prosodic structures which require more than linear time for processing.

So let’s have a look.

The second case: centre-embedding?

(8) First condition: A while [B and C]

{Why does Anna think that craftsmen have more expensive cars than musicians?}

[Weil der Maler einen Jaguar hat]$_A$, [[während die Sängerin einen Lada besitzt]

und [der Geiger einen Wartburg fäh

'Because the painter has a Jaguar, while

violinist drives a Wartburg.'

(9) Second condition: [A and B] while C

{Why does Anna think that musicians have

[[Weil die Sängerin einen Lada besitzt]$_A$, [und der Geiger einen Wartburg fährt]$_B$],

[während der Maler einen Jaguar hat]$_C$

'Because the singer possesses a Lada, and the violinist drives a Wartburg, while the painter has a Jaguar.'

Comment:

No, linear prosody because the trees are right-branching and left-branching. One would expect a finite state intonation grammar to handle them.

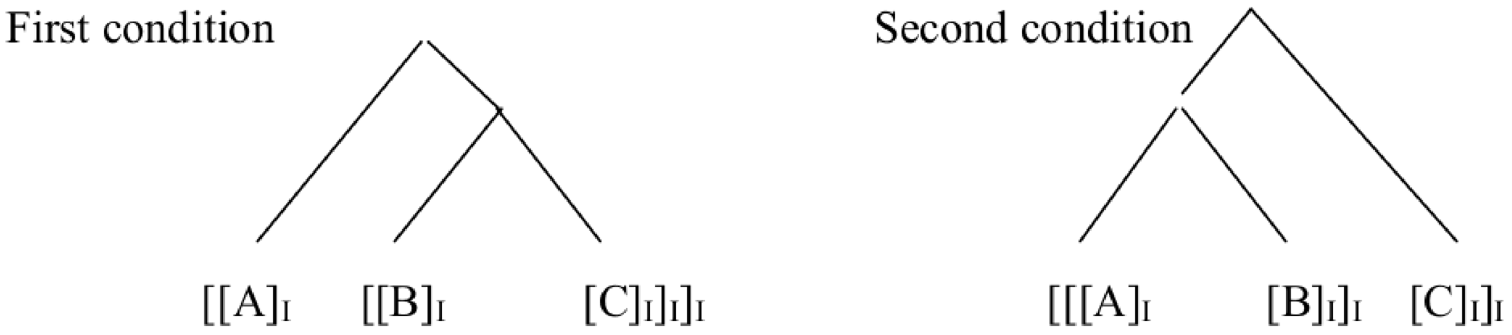
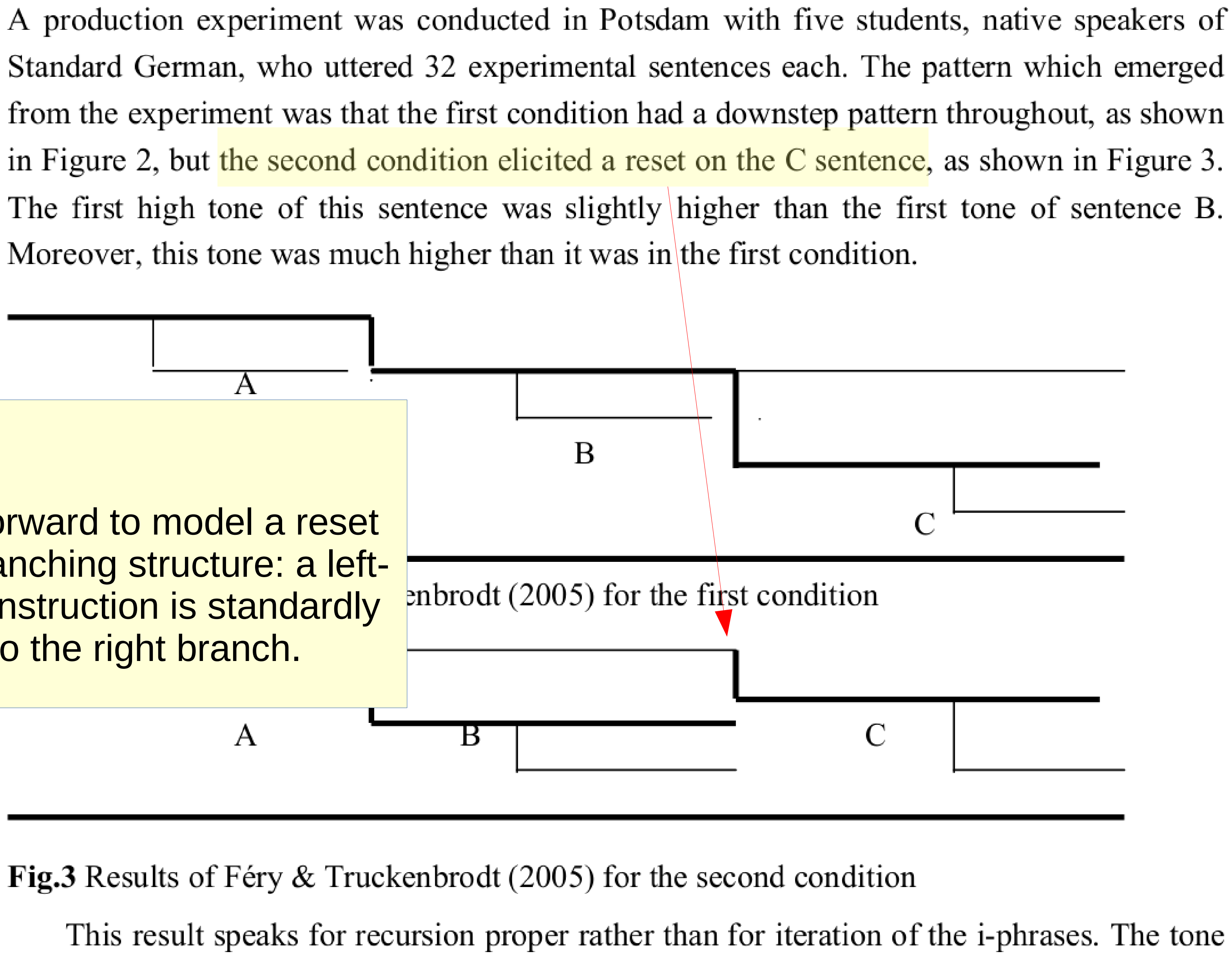


Fig.1 Two conditions in the experiment reported in Féry & Truckenbrodt (2005)

Féry, Caroline. 2010. Recursion in prosodic structure. *Linguistics*.

The second case: centre-embedding?



A production experiment was conducted in Potsdam with five students, native speakers of Standard German, who uttered 32 experimental sentences each. The pattern which emerged from the experiment was that the first condition had a downstep pattern throughout, as shown in Figure 2, but the second condition elicited a reset on the C sentence, as shown in Figure 3. The first high tone of this sentence was slightly higher than the first tone of sentence B. Moreover, this tone was much higher than it was in the first condition.

enbrodt (2005) for the first condition

Fig.3 Results of Féry & Truckenbrodt (2005) for the second condition

This result speaks for recursion proper rather than for iteration of the i-phrases. The tone

Comment

It is straightforward to model a reset with a left-branching structure: a left-branching construction is standardly subordinate to the right branch.

Note: recursion is not always RECURSION

Summary:

There is a common misunderstanding that *right-branching and left-branching trees are centre-embedding, even if the non-terminal symbols are repeated and thus apparently ‘included’.*

This is false!

In right and left branching, the apparently embedded items are simply added on at the end, resulting in a linear, iterative pattern.

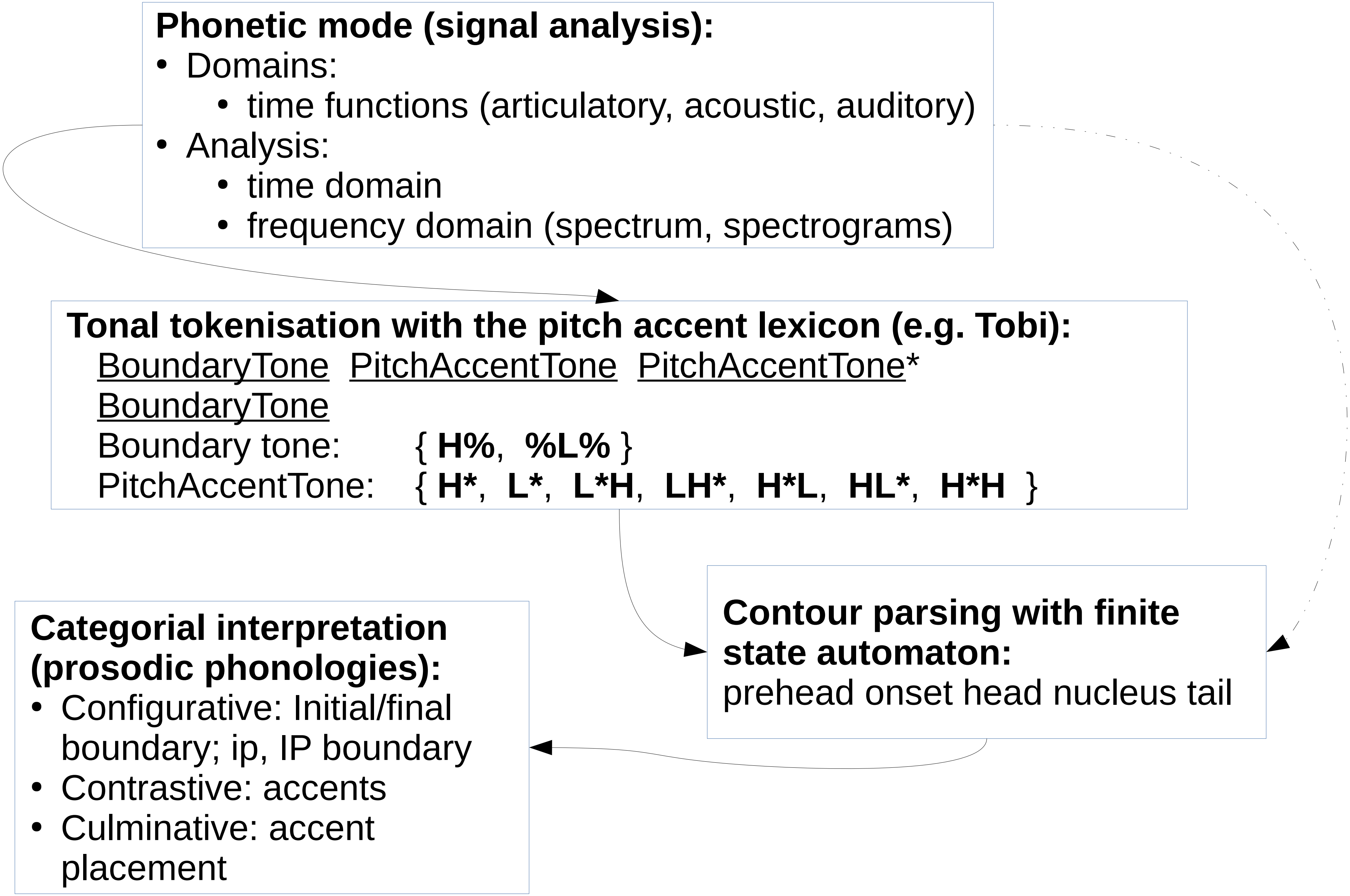
Centre-embedding (self-embedding) structures are generated by Type 2 context-free phrase structure grammars (also Type 1, Type 0 grammars), and require polynomial or exponential time and space relative to the length of the input.

Right-branching and left-branching structures are generated by Type 3 regular or linear trammars and require linear time and finite space.

Claim:

Prosodic patterns in spontaneous speech are not centre-embedding and are generated by Type 3 grammars (or accepted by finite state automata).

A computational side note



Phonetic mode (signal analysis):

- Domains:
  - time functions (articulatory, acoustic, auditory)
- Analysis:
  - time domain
  - frequency domain (spectrum, spectrograms)

Tonal tokenisation with the pitch accent lexicon (e.g. Tobi):

BoundaryTone PitchAccentTone PitchAccentTone* BoundaryTone

Boundary tone: { **H%**, **%L%** }

PitchAccentTone: { **H***, **L***, **L*H**, **LH***, **H*L**, **HL***, **H*H** }

Contour parsing with finite state automaton:
prehead onset head nucleus tail

Categorial interpretation (prosodic phonologies):

- Configurative: Initial/final boundary; ip, IP boundary
- Contrastive: accents
- Culminative: accent placement

End