Prosody: speech rhythms and melodies

5. The Prosody of Sentences and Words

Dafydd Gibbon

Summer School
Contemporary Phonology and Phonetics
Tongji University 9-15 July 2016

The syntax (= structure) of prosody

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

Vocabulary (Lexicon, Dictionary, Inventory)

- List of items (phonemes, morphemes, words, idioms, …)
- Set of paradigmatic (classificatory, similarity) relations

Constructor (Rule system, Constraint system)

- Generator / Parser (creation and analysis of structures)
- Set of syntagmatic (compositional) relations

The syntax (= structure) of prosody

- Compositional operations in prosody:
  - Sequencing:
    - concatenation of tokens (cf. standard phonologies & grammars)
  - Parallelism:
    - synchronisation; overlap (cf. autosegmental phonology)
  - Grouping:
    - generalisation; domain (cf. metrical phonology)
- These operations are interpreted in terms of <u>temporal relations</u>

Formal Foundations of Prosody: Event logics

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

  Precedence: $A \prec B$

  Immediate Precedence: A ^ B

  Overlap: $A \circ B$

  Include: $A \supset B$

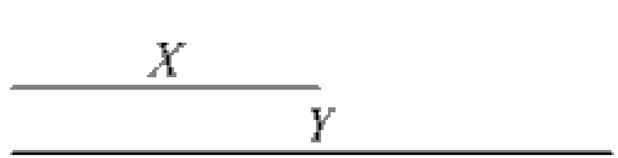
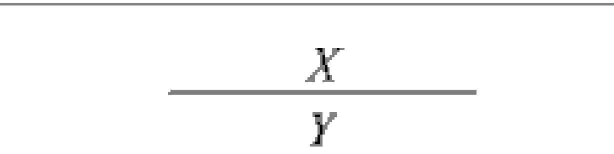
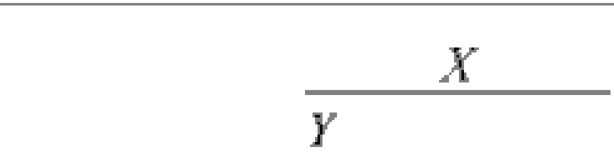
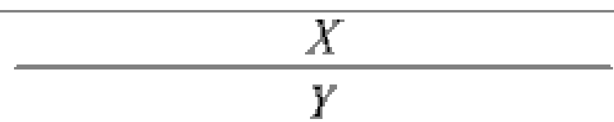
Ontological decision:

- points?
- intervals?

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

Event Phonology (Steven Bird; Julie Carson-Berndsen)

Formal Foundations of Prosody: Allen's Interval Algebra

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

An apparently simple question:

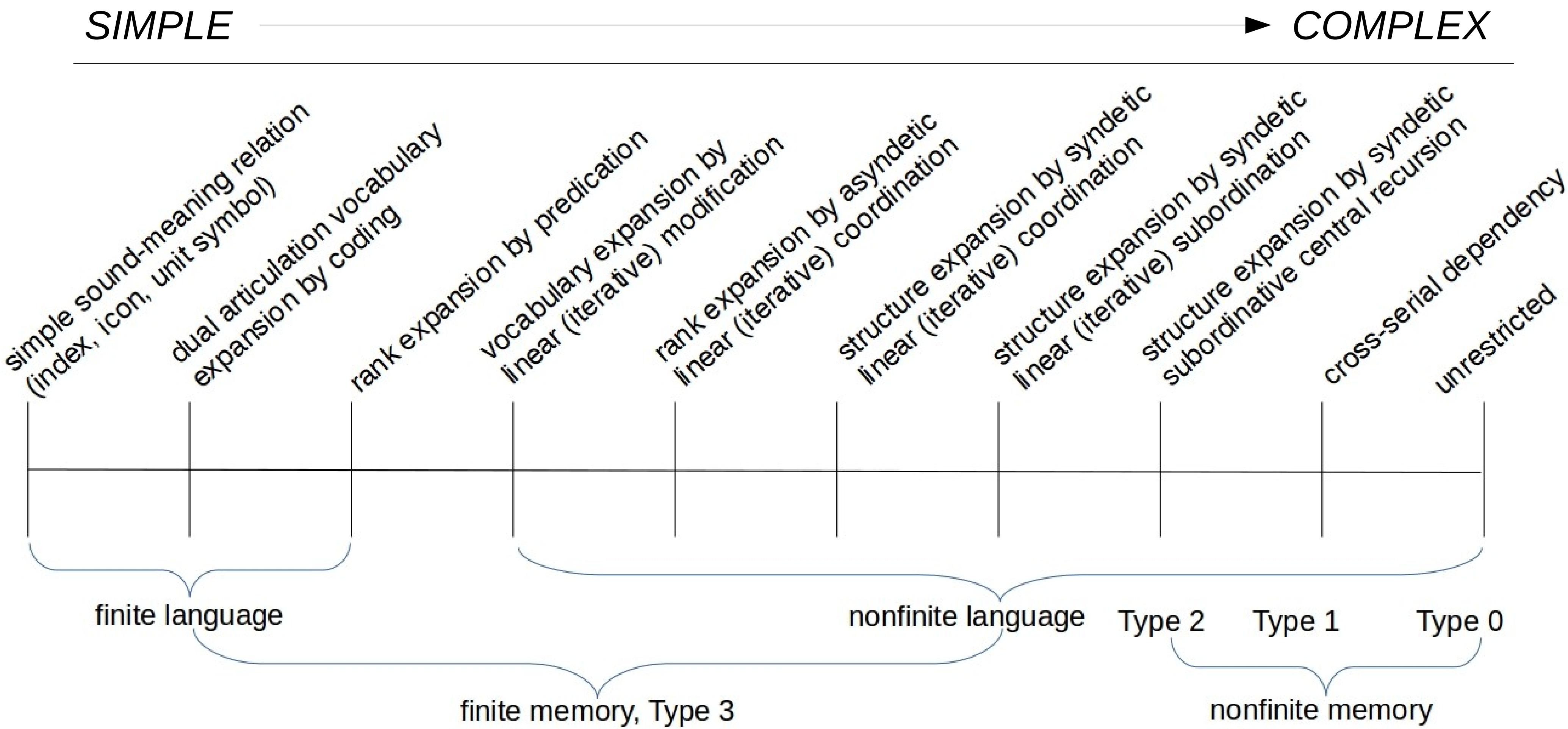
IF PROSODY MARKS GRAMMATICAL STRUCTURES,

CAN PROSODY MARK RECURSION?

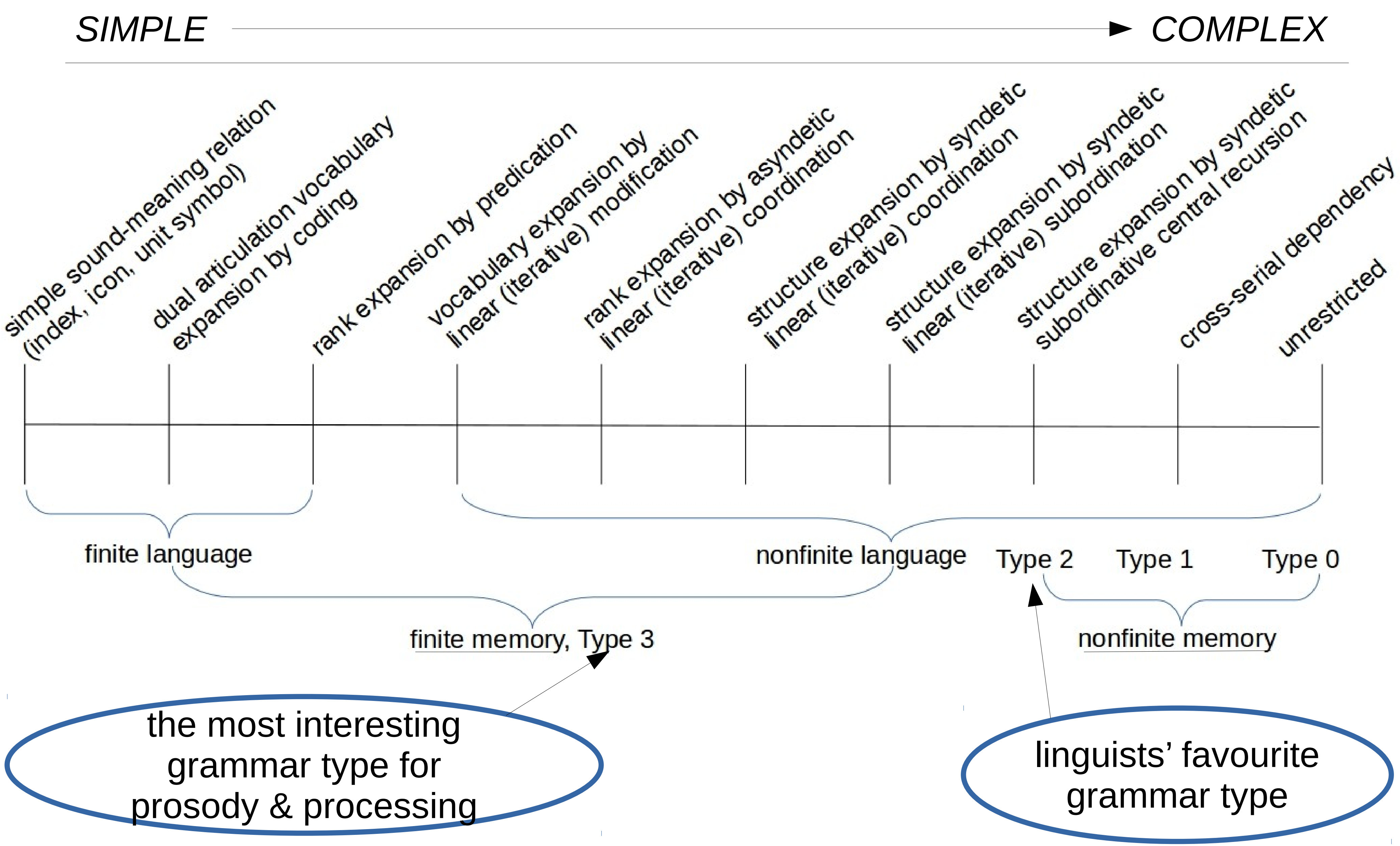
First: sequences

and a scale of formal grammars

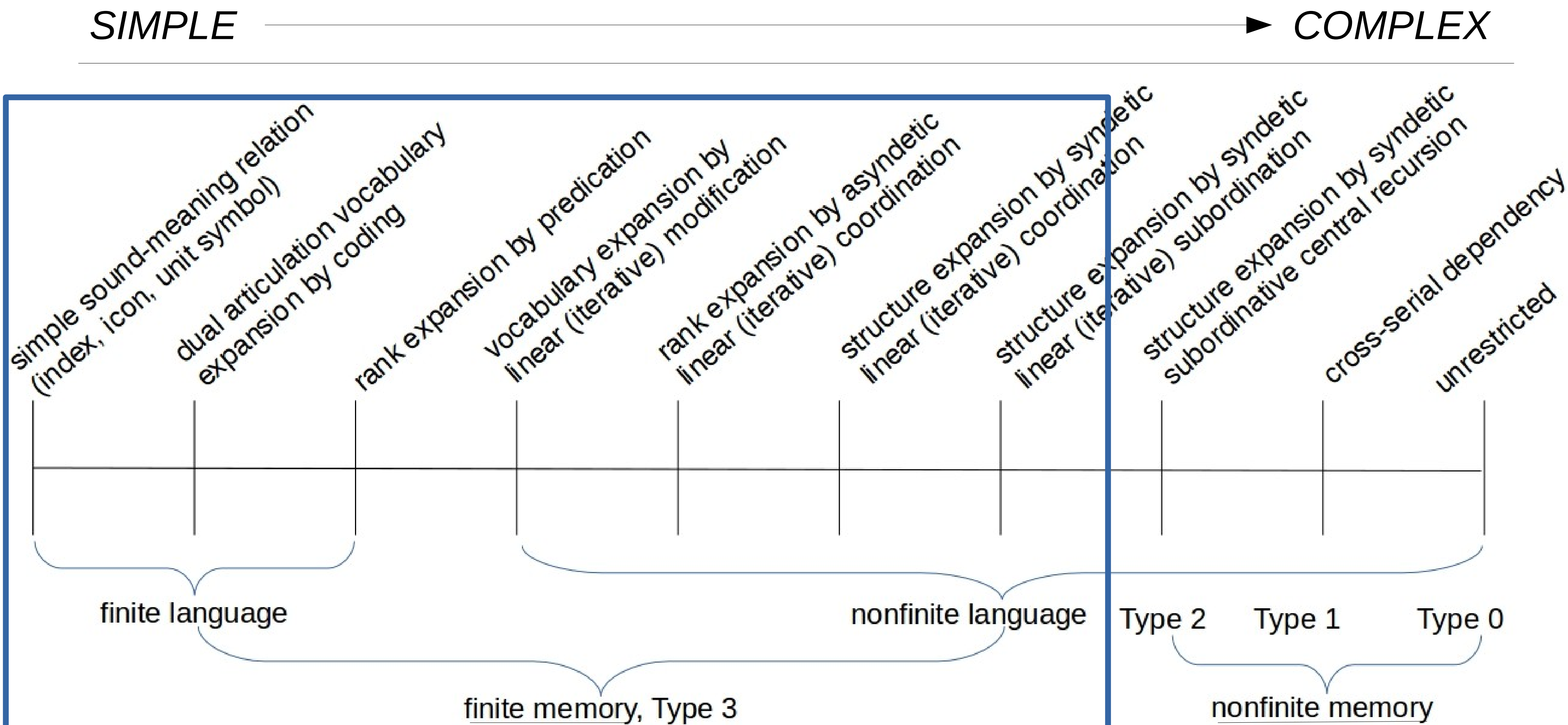
A complexity scale of formal grammars



A complexity scale of formal grammars



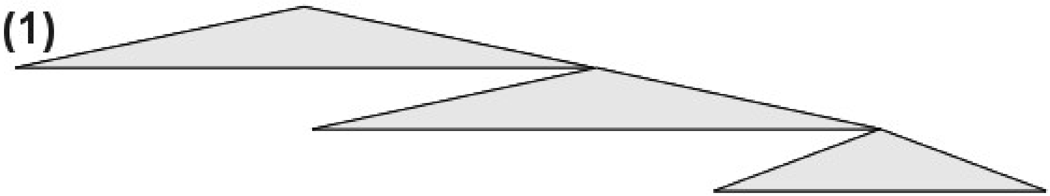
A complexity scale of formal grammars



THE STORY Of THE ACQUISITION AND THE EVOLUTION OF GRAMMAR?

Recursivity and the hierarchy of formal grammars

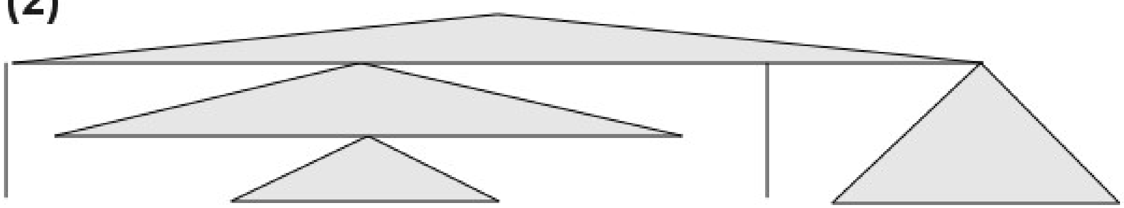
(1)



This is the dog that chased the cat that ate the mouse …

Right-branching linear recursion / iteration.

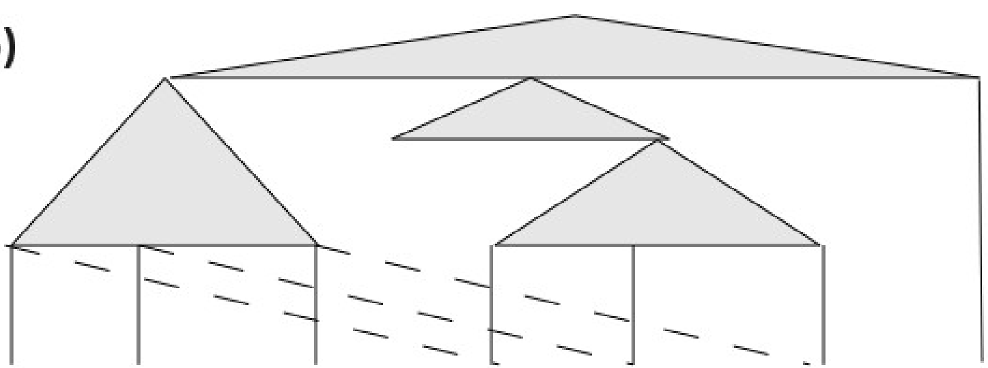
(2)



If the man who John met goes home then Jane will smile

Centre-embedding hierarchical recursion.

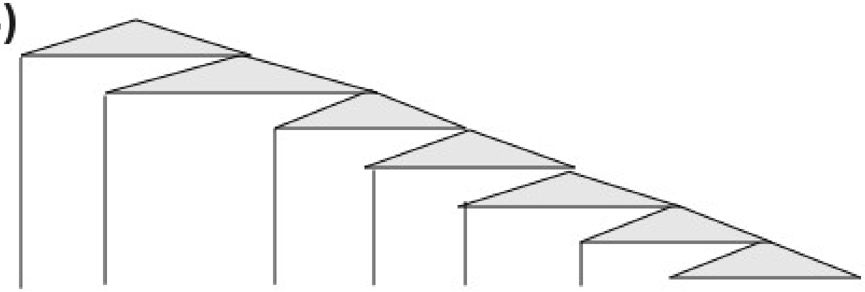
(3)



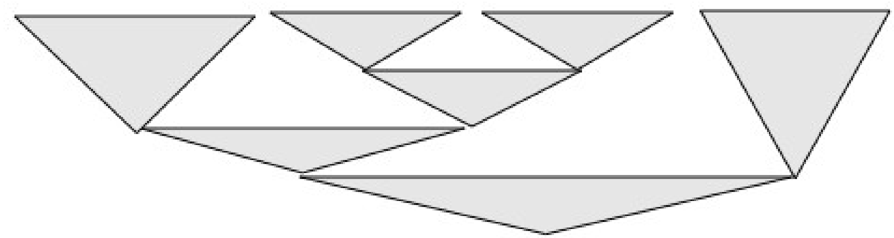
June, Jane and Jean love Mick, Dick and Nick, respectively

Recursive cross-serial dependency.

(4)



twin cylinder over head cam shaft motor bike



Mismatch between linear syntagmatic and hierarchical hermeneutic recursion.

Confusion in the ‘recursion’ / ‘merge’ discussion!

Note that hierarchies *per se* are defined recursively at an abstract level, but they do not necessarily represent recursivity!

Confusion in the ‘recursion’ / ‘merge’ discussion!

- A general definition of a branching structure is recursive in the mathematical sense:
  - branching nodes

        dominate branching nodes

            dominate branching nodes …

  - until leaf nodes are reached

Confusion in the 'recursion' / 'merge' discussion!

- Not every branching structure in linguistics is recursive in this mathematical sense:

  If a symbol in a tree recurs lower down in the tree

  - then the tree is <u>recursive</u> and may be <u>arbitrarily deep</u> and a set of such trees in principle requires infinite memory
  - otherwise the tree is <u>not recursive</u>, <u>fixed finite depth</u> and only requires finite memory:

    the <u>Prosodic Hierarchy</u> with the <u>Strict Layering Hypothesis</u>

    simple sentences and simple phrases

    syllables

    ...

Confusion in the 'recursion' / 'merge' discussion!

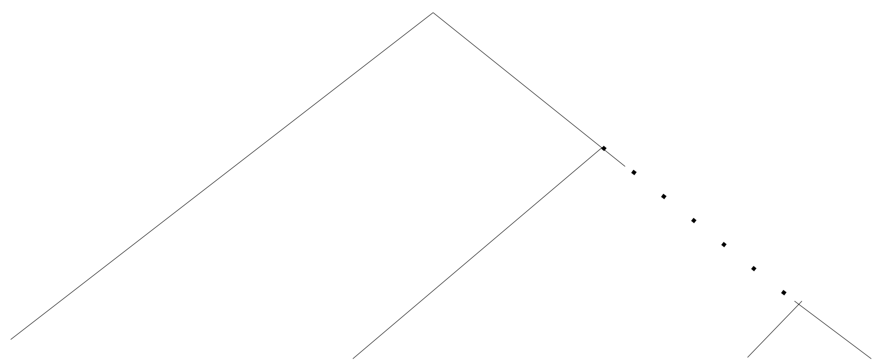
- Grammars which only require finite memory generate
  - either non-recursive trees of finite depth
  - unilaterally right or left branching recursive trees

can easily be modelled as 'flat grammar' by means of finite state automata

S → john VP

VP → laughed

VP → said that S



Confusion in the ‘recursion’ / ‘merge’ discussion!

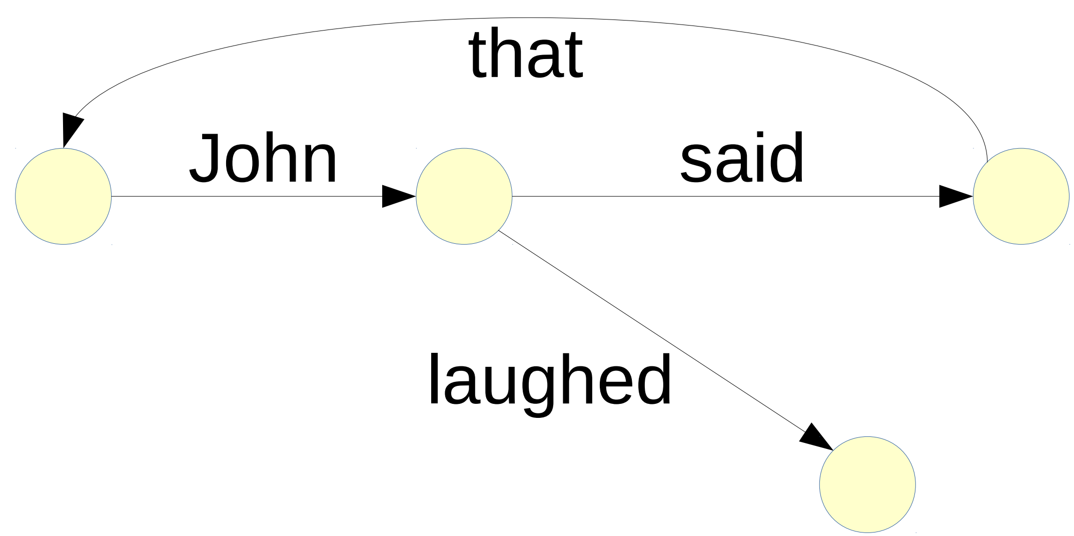
- Grammars which only require finite memory generate
  - either non-recursive trees of finite depth
  - unilaterally right or left branching recursive trees

and can easily be modelled as ‘flat grammar’ by means of finite state machines

S → john VP

VP → laughed

VP → said that S



Two main kinds of recursion

- Linear recursion (left <u>or</u> right branching, not both)

  {*the car*, *Jim's car*, *Jim's dad's car*, *Jim's dad's mate's car*, ...}

  Left-branching: A → B *car*, B → B {dad'*s*, mate'*s*}, B → {*the*, Jim'*s*}

  Right-branching: A → {*the*, Jim'*s*} B, B → {dad'*s*, *mate's*} B, B → *car*

  - Equivalent to iteration (flat recursion):
    - Jim's (dad's, mate's)* car

Two main kinds of recursion

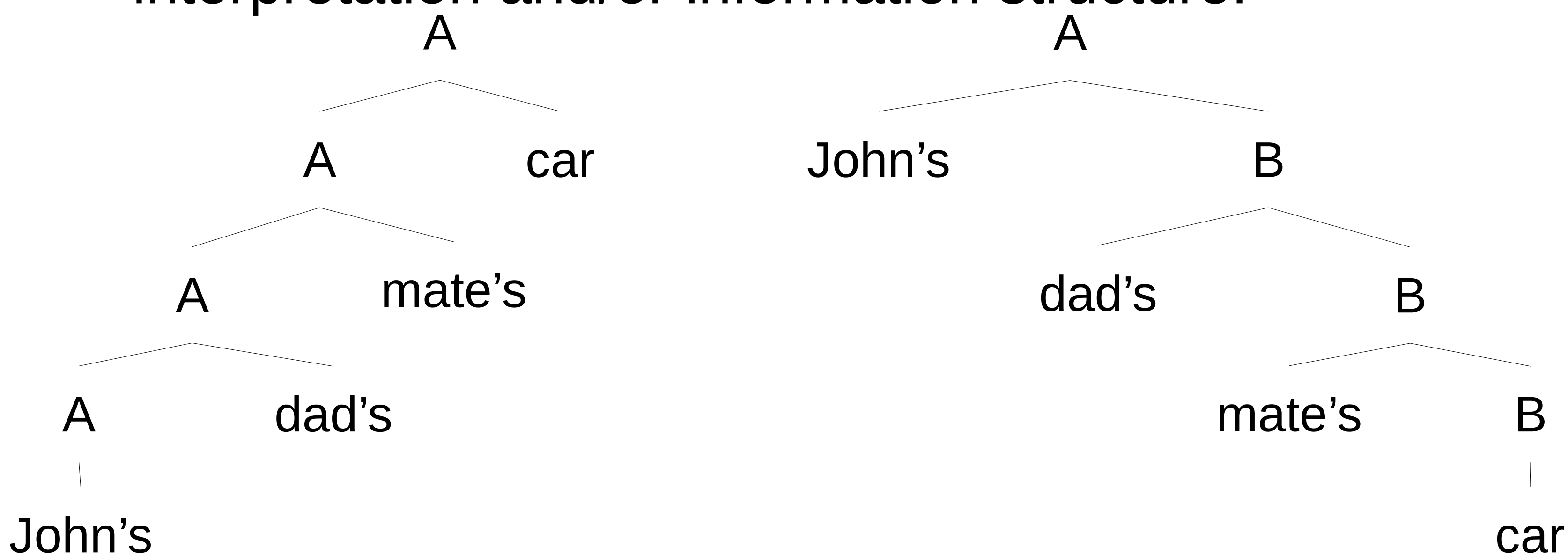
- Linear recursion (left <u>or</u> right branching, not both)

  {*the car*, *Jim’s car*, *Jim’s dad’s car*, *Jim’s dad’s mate’s car*, ...}

  Left-branching: A → B *car*, B → B {dad*’s*, mate*’s*}, B → {*the*, Jim*’s*}

  Right-branching: A → {*the*, Jim*’s*} B, B → {dad*’s*, *mate’s*} B, B → *car*

  - Equivalent to iteration (flat recursion):
    - Jim’s (dad’s, mate’s)* car
  - Tree structures are not necessary, but helpful for semantic interpretation and/or information structure:



Two main kinds of recursion

- Linear recursion (left <u>or</u> right branching, not both)

{*the* ... *ate's car*, ...}

→ {*the*, Jim*'s*}

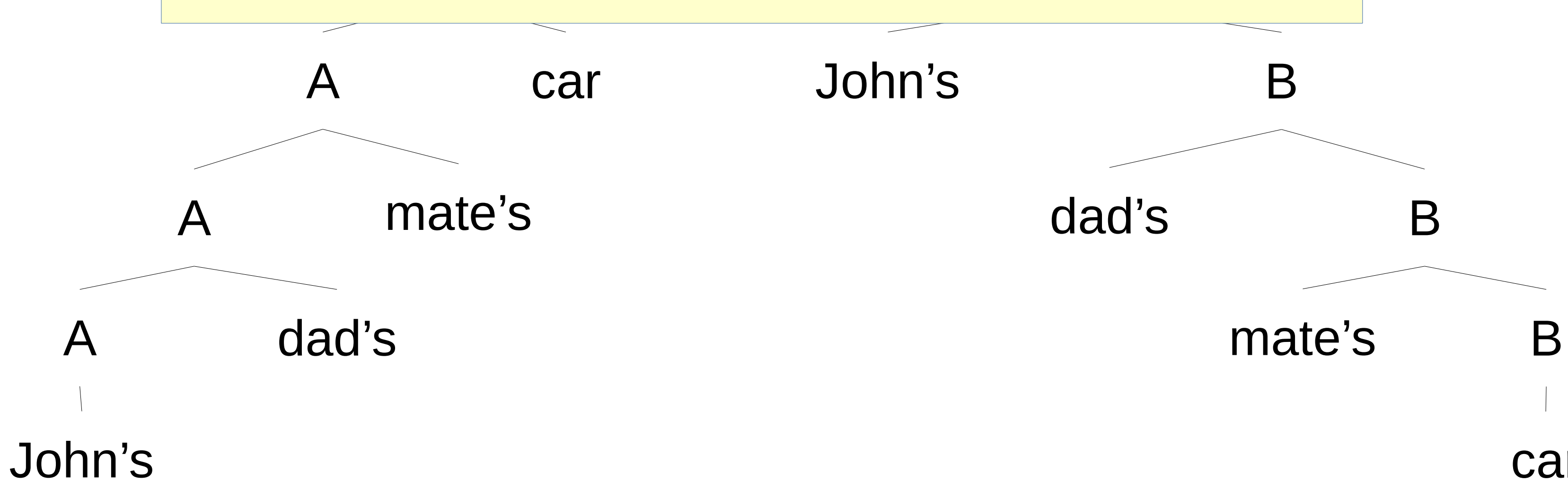
ate's} B, B → *car*

- E...
- T... for semantic in...

Unilaterally branching trees of arbitrary depth are not a problem for prosodic marking.

They are equivalent to flat recursion / iteration and can be represented by:
- sequence of phrases
- with breaks
- with final nucleus

Unilaterally branching trees cornform to the *Strict Layering Hypothesis*.



Two main kinds of recursion

- Linear recursion (left <u>or</u> right branching, not both)

{*the* … *ate's car*, ...}

… → {*the*, Jim'*s*}

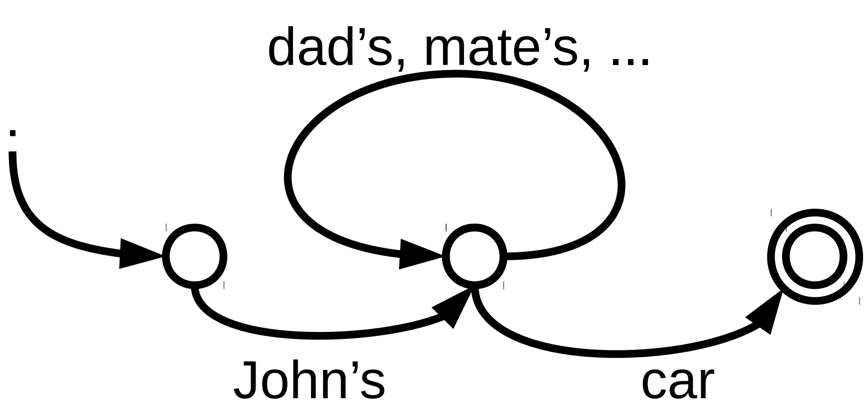
… *ate's*} B, B → *car*

- E…
- T… for semantic
  i…

Unilaterally branching trees of arbitrary depth are not a problem for prosodic marking.

They are equivalent to flat recursion / iteration and can be represented by:
- a sequence of phrases
- with breaks
- with final nucleus

Unilaterally branching trees cornform to the *Strict Layering Hypothesis*.



This simple grammar, a finite state machine represented as a transition diagram, is compatible with both left and right branching grammars

Two main kinds of recursion

- Centre-embedding recursion has different properties:
  - Logical centre-embedding:
    - if - then
    - (why -) because
  - Descriptive centre-embedding:
    - relative clauses (restrictive, non-restrictive)
      - The man whose brother, who married Jane, is a doctor is a teacher.
  - Declarative centre-embedding:
    - Indirect speech:
      - That what I said is true is obvious.
  - Parenthetic centre-embedding:
    - Rosie’s birthday, by the way, was last Tuesday.
    - Last Tuesday, which, by the way, was Rosie’s birthday, I left.

Two main kinds of recursion

- Centre-embedding recursion
  - is rarely necessary at the level of language forms: replaceable by a linear sequence of flat forms with pointers – delegated to semantics and thus to general cognitive processes

Try to find an intonation which marks the structure of this sentence!

If, if it rains tomorrow then we'll visit the museum, then, if it rains the day after then we'll go to the art gallery, ok?

Two main kinds of recursion

- Centre-embedding recursion:
  - is rarely necessary at the level of language forms: replaceable by a linear sequence of flat forms with pointers – delegated to semantics and thus to general cognitive processes

Try to find an intonation which marks the structure of this sentence!

If, as you say, if it rains tomorrow then we’ll visit the museum, then, please listen closely, if it rains the day after then we’ll go to the art gallery, ok?

Two main kinds of recursion

- Centre-embedding recursion:
  - is rarely necessary at the level of language forms: replaceable by a linear sequence of flat forms with pointers – delegated to semantics and thus to general cognitive processes

Try to find an intonation which marks the structure of this sentence!

If, as you say,

if it rains tomorrow then we'll visit the museum,

then, please listen closely,

if it rains the day after then we'll go to the art gallery, ok?

a "structure-marking" strategy

Two main kinds of recursion

- Centre-embedding recursion:
  - is rarely necessary at the level of language forms: replaceable by a linear sequence of flat forms with pointers – delegated to semantics and thus to general cognitive processes

Try to find an intonation which marks the structure of this sentence!

You said, if it rains tomorrow we'll visit the museum. So if it rains the day after, we'll go to the art gallery, ok?

a “de-embedding” strategy

Two main kinds of recursion

- Centre-embedding recursion:
  - is rarely necessary at the level of language forms: re… s with p… o general c…

Tr… sentence!

…m. So if it rains the day after, we'll go to the art gallery, ok?

Centre-embedded trees of arbitrary depth are a real problem for prosodic marking, which only works to a depth of about 2 or 3.

This is not an accident, and affects more than prosody.

Even with the memory enhancement of written language, centre-embedded constructions with depth more than 2 or 3 are very difficult to understand.

a "de-embedding" strategy

Two main kinds of recursion

- In fact marking any kind of hierarchy with prosody is a problem, beyond depth 2 or 3
    - stress levels are usually limited to 2 or 3 (primary, secondary, unstressed)
    - Bierwisch and others criticised unlimited derivation of stress levels from generative gramar hierarchies:

the man in the car saw Mary

An apparently simple question:

IF PROSODY MARKS GRAMMATICAL STRUCTURES,

CAN PROSODY MARK RECURSION?

The answer:

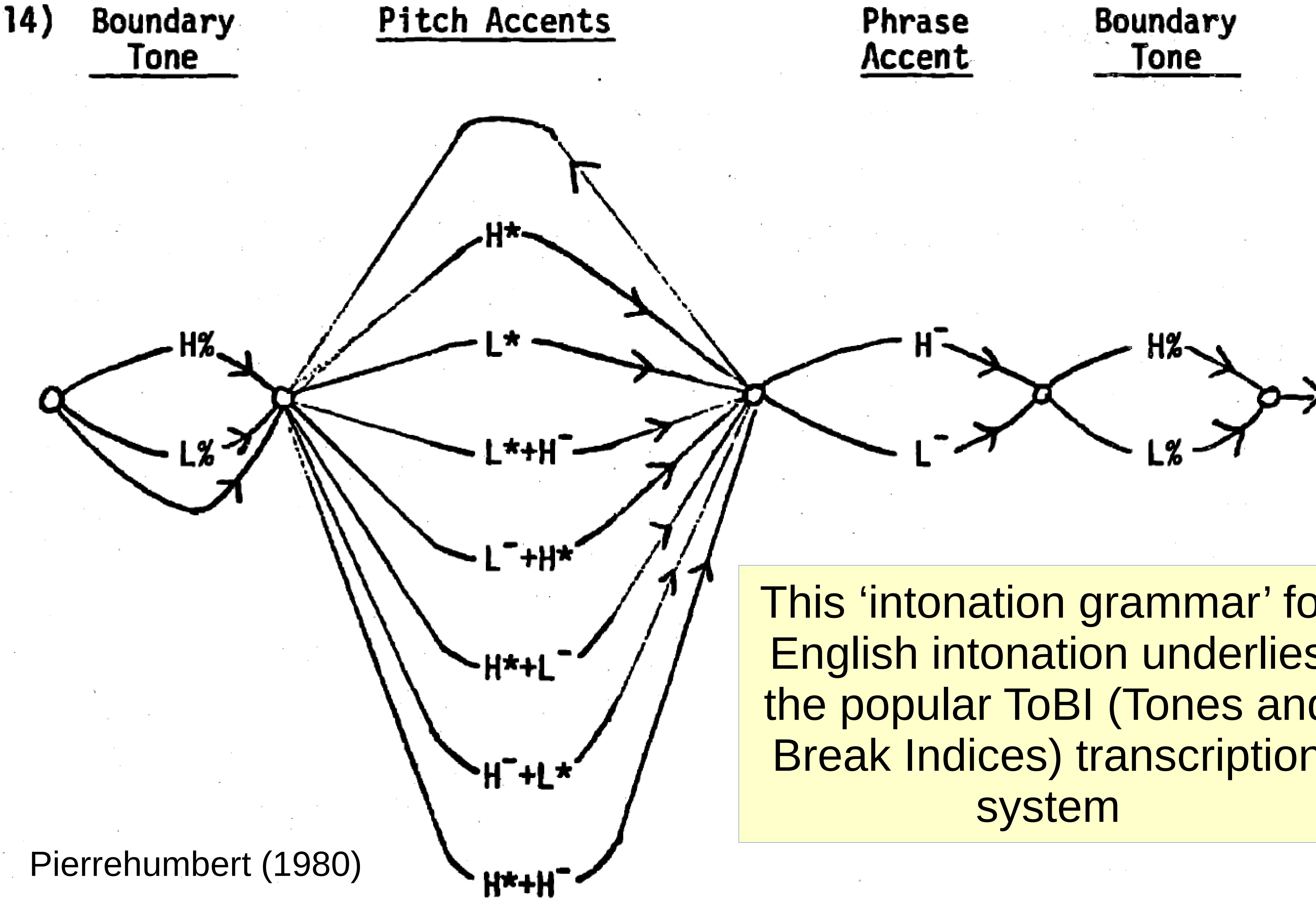
FLAT, ITERATIVE RECURSiON – NO PROBLEM.

CENTRE-EMBEDDED RECURSION – LIMITED DEPTH

Syntagmatic structure of English intonation:

Pierrehumbert's Finite Machine Model

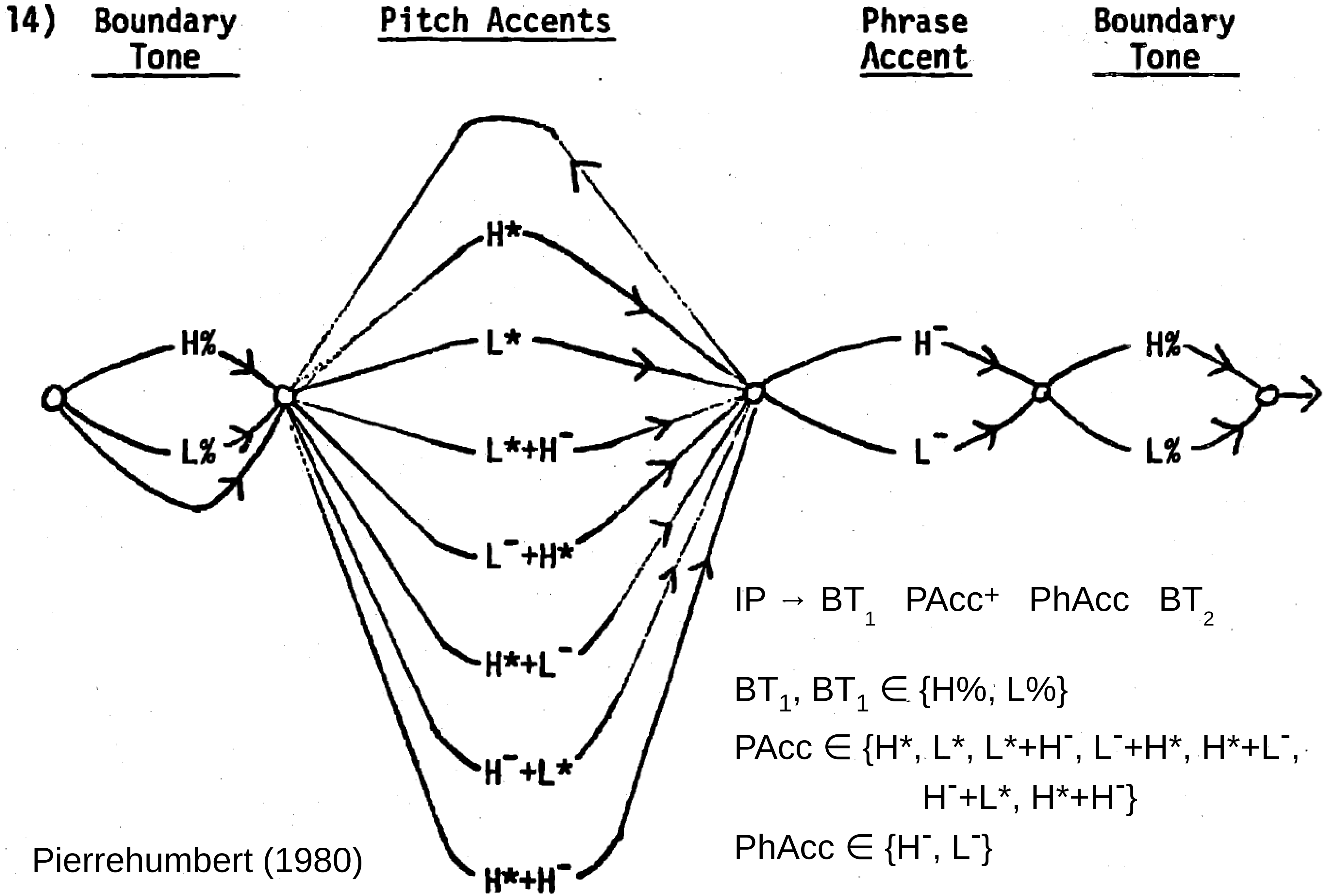
Syntagmatic structure: a Finite Machine Model



This ‘intonation grammar’ for English intonation underlies the popular ToBI (Tones and Break Indices) transcription system

Pierrehumbert (1980)

Syntagmatic structure: a Finite Machine Model



$IP \rightarrow BT_1 \quad PAcc^+ \quad PhAcc \quad BT_2$

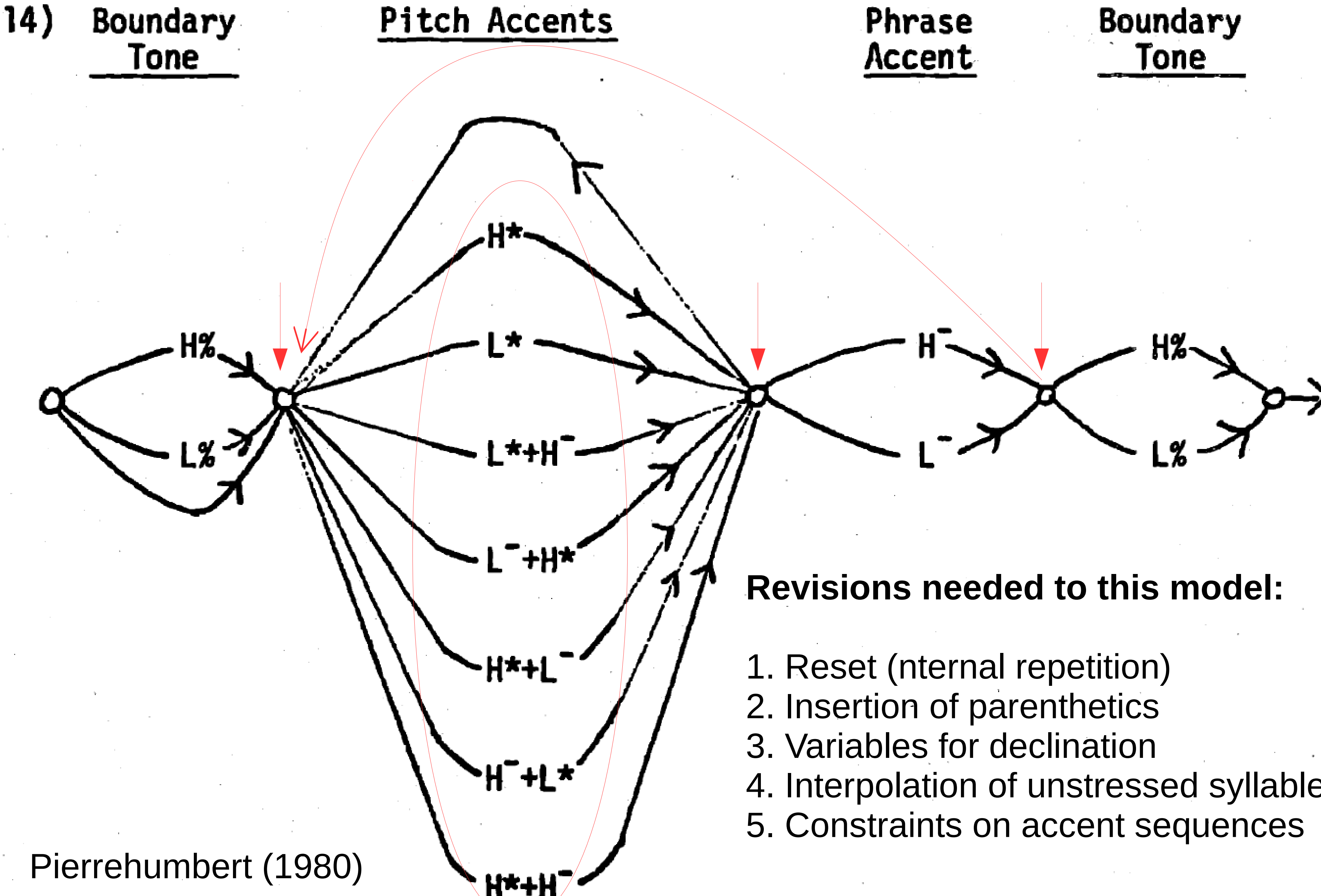
$BT_1, BT_1 \in \{H\%, L\%\}$

$PAcc \in \{H^*, L^*, L^*{+}H^-, L^-{+}H^*, H^*{+}L^-,$
$H^-{+}L^*, H^*{+}H^-\}$

$PhAcc \in \{H^-, L^-\}$

Pierrehumbert (1980)

Syntagmatic structure: a Finite Machine Model



Revisions needed to this model:

1. Reset (nternal repetition)
2. Insertion of parenthetics
3. Variables for declination
4. Interpolation of unstressed syllables
5. Constraints on accent sequences

Pierrehumbert (1980)

The finite depth grammar of the Prosodic Hierarchy

Prosodic Category inventory:

PC = {Utt, IP, PhP, PrWd, omega, Ft phi, syll, mora, segment}

Prosodic Hierarchy ordering:

L = <Utt, IP, PhP, PrWd, omega, Ft phi, syll, mora, segment>

l1 = Utt, l2 = IP, … l9 = segment

Structural constraints on Pros

But iterative recursion at the same rank is ok.

Strict Layering Hypothesis:

PC at L_i dominates only PCs at L_{i+1}!

- Fixed depth (no recursion): No PC at L_i dominates a PC at L_{i+1}
- Exhaustivity: All PCs at L_i are dominated by a single PC at L_{i-1}

Headedness:

- Every PC at L_i immediately dominates a PC at L_{i+1}

A formal note on two main kinds of recursion

(a popular topic these days)

from the point of view of a computational linguist

A formal note on two main kinds of recursion

- In fact marking any kind of hierarchy with prosody is a problem, beyond depth 2 or 3
  - stress levels are usually limited to 2 or 3 (primary, secondary, unstressed)
  - Bierwisch and others criticised unlimited derivation of sentence and word stress levels from generative gramar hierarchies:

4	3	4	5	2	3	1
the	man	in	the	car	saw	Mary

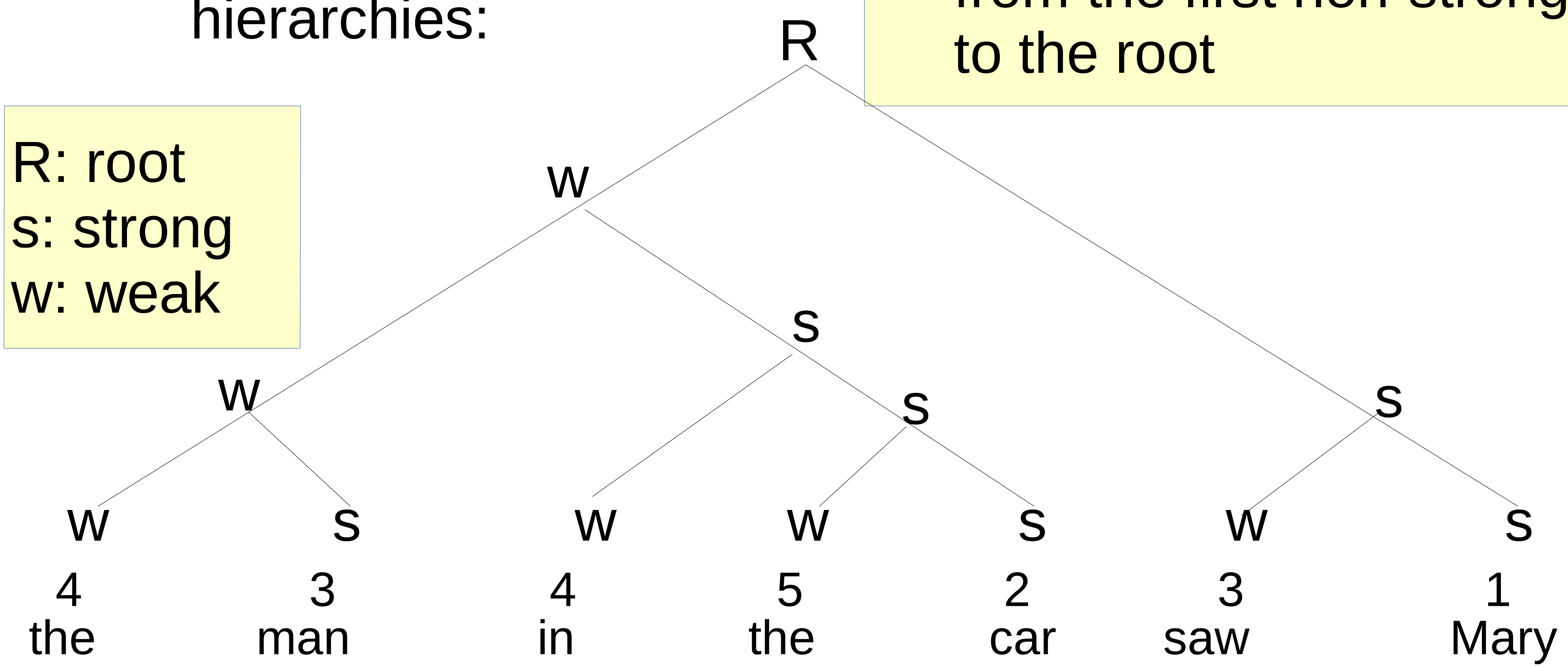
A formal note on two main kinds of recursion

- In fact marking any kind
  prosody is a problem, be
  - stress levels are usually lir
    secondary, unstressed)
  - Bierwisch and others critic
    sentence and word stress
    hierarchies:

Liberman's bottom-up algorithm for the Nuclear Stress Rule:

for each leaf in the tree:
  stress level =
    number of nodes in the path
    from the first non-strong node
    to the root

R: root
s: strong
w: weak



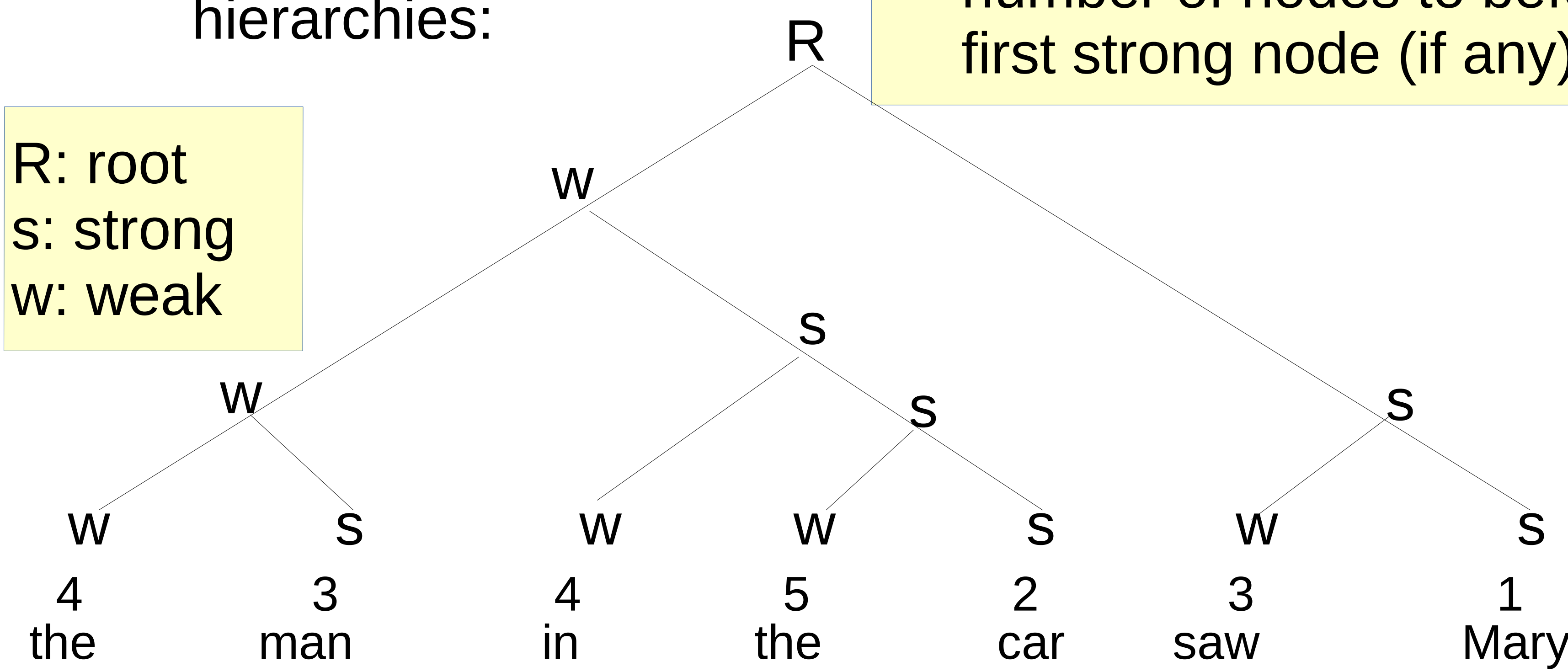
A formal note on two main kinds of recursion

- In fact marking any kind
prosody is a problem, be
  - stress levels are usually lir
secondary, unstressed)
  - Bierwisch and others critic
sentence and word stress
hierarchies:

Equivalent top-down algorithm for the Nuclear Stress Rule:

starting at the root:

for each path to a leaf:
stress level =
number of nodes to before the
first strong node (if any)



A formal note on two main kinds of recursion

- In fact marking any kind
  prosody is a problem, be
  - stress levels are usually lin
    secondary, unstressed)
  - Bierwisch and others critic
    sentence and word stress
    hierarchies:

Equivalent bracket-counting left-right algorithm for the Nuclear Stress Rule:

```
set counter to 1:
if item is left bracket:
   counter = counter + 1
if item is right bracket:
   counter = counter -1
if item is leaf:
   if previous item = left bracket:
       stress = counter
    if next item = right bracket:
       stress = counter - 1
```

R: root
s: strong
w: weak

(((the man) (in (the car))) (saw Mary))

A formal note on two main kinds of recursion

- In fact marking any kind
  prosody is a problem, be
  - stress levels are usually lir
    secondary, unstressed)
  - Bierwisch and others critic
    sentence and word stress
    hierarchies: and others crit
    stress levels from generati

R: root
s: strong
w: weak

Equivalent bracket-counting left-right algorithm for the Nuclear Stress Rule:

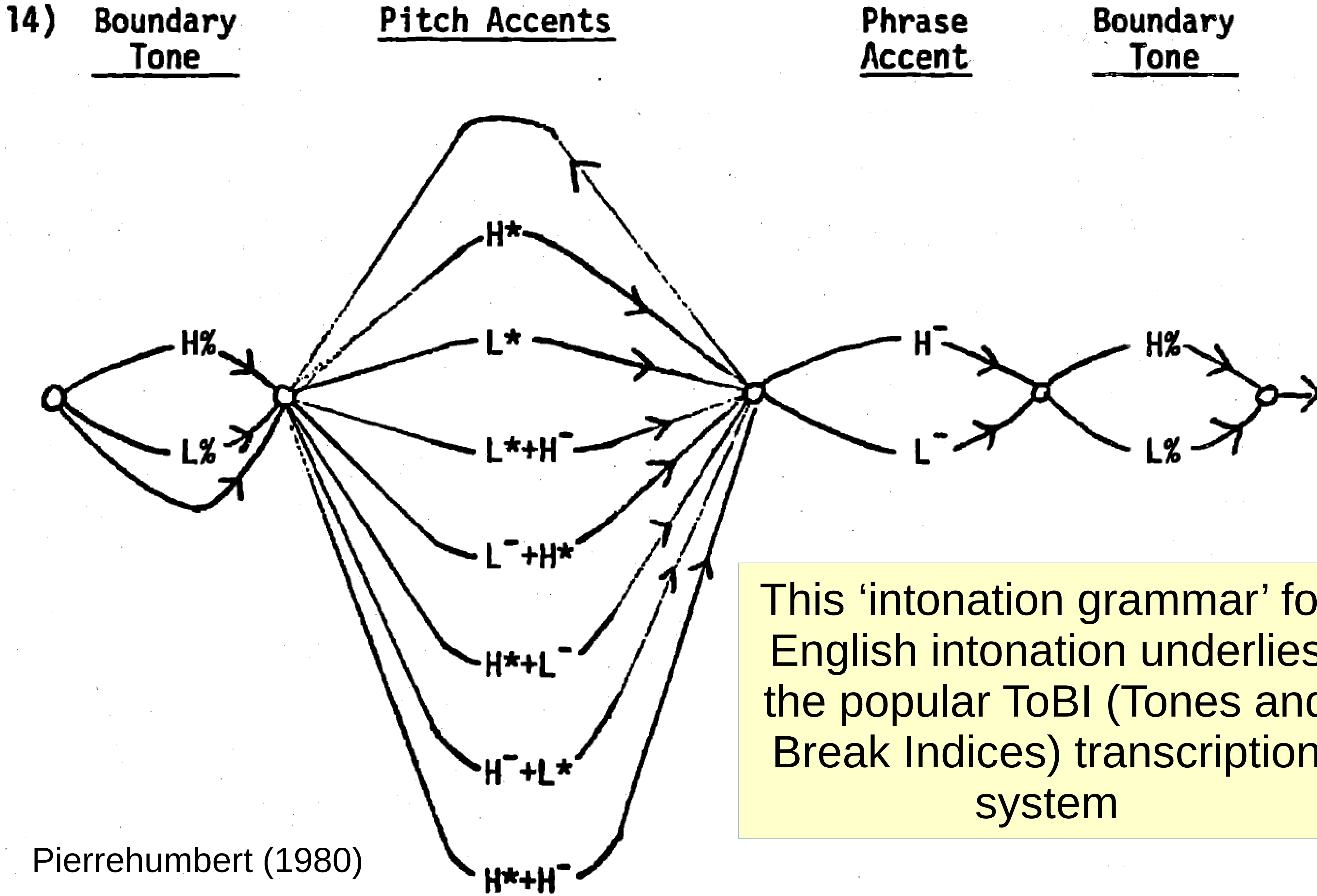
```
set counter to 1:
if item is left bracket:
   counter = counter + 1
if item is right bracket:
   counter = counter -1
if item is leaf:
   if previous item = left bracket:
       stress = counter
    if next item = right bracket:
        stress = counter - 1
```

4	3	4	5	2	3	1
(((the	man)	(in	(the	car)))	(saw	Mary))

Syntagmatic structure of English intonation:

Pierrehumbert's Finite Machine Model

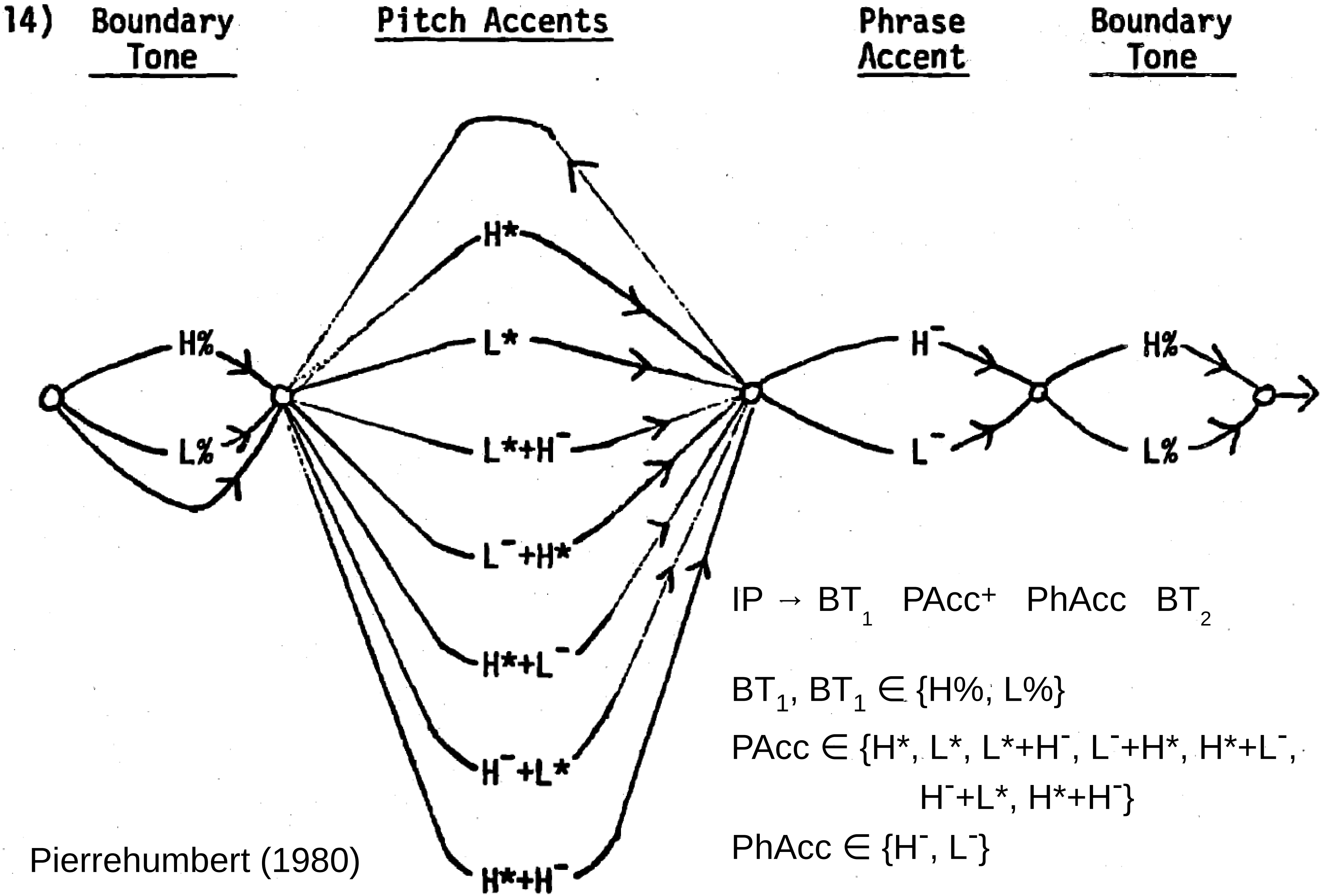
Syntagmatic structure: a Finite Machine Model



This ‘intonation grammar’ for English intonation underlies the popular ToBI (Tones and Break Indices) transcription system

Pierrehumbert (1980)

Syntagmatic structure: a Finite Machine Model



$IP \rightarrow BT_1 \quad PAcc^+ \quad PhAcc \quad BT_2$

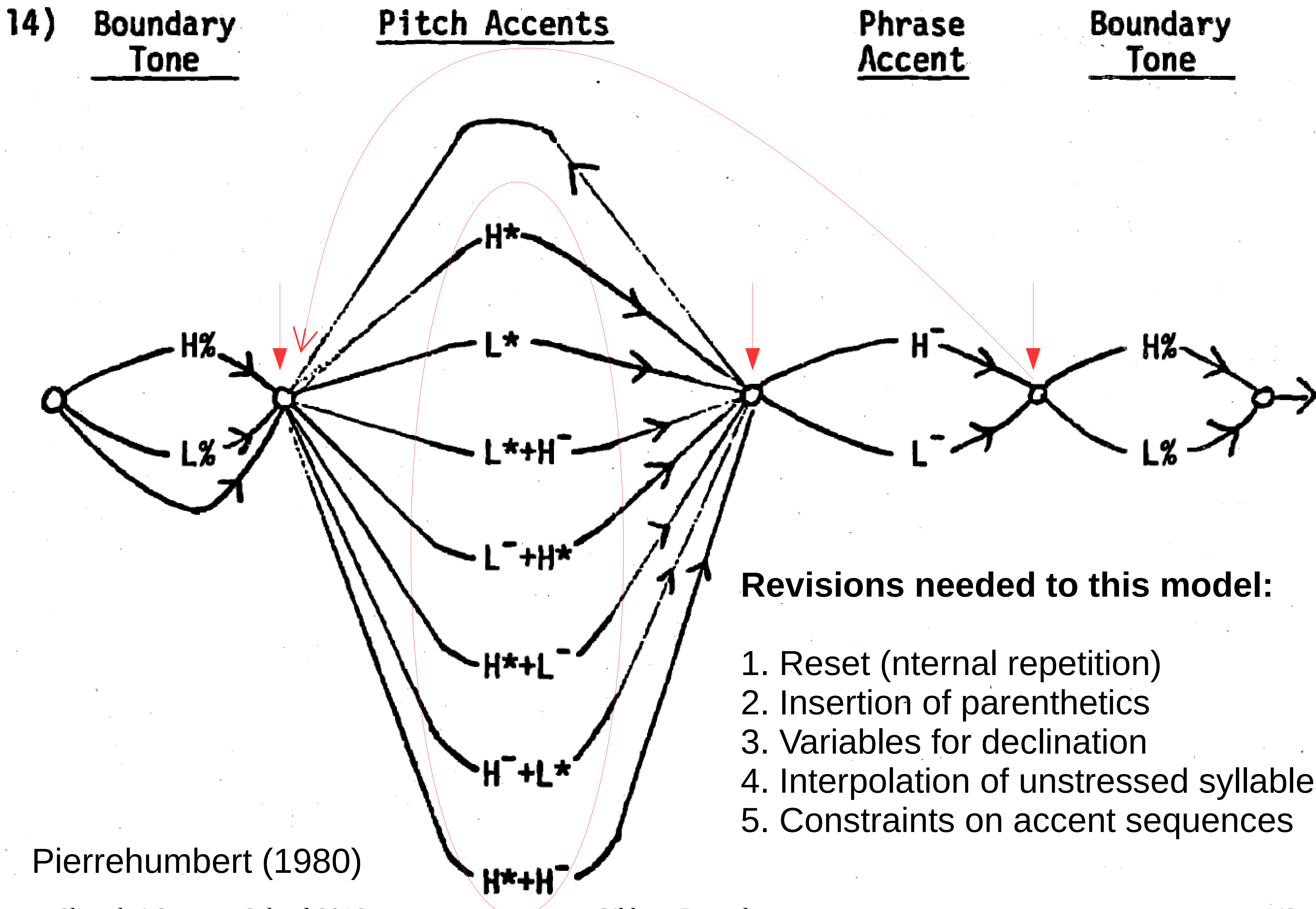
$BT_1, BT_1 \in \{H\%, L\%\}$

$PAcc \in \{H^*, L^*, L^*+H^-, L^-+H^*, H^*+L^-, H^-+L^*, H^*+H^-\}$

$PhAcc \in \{H^-, L^-\}$

Pierrehumbert (1980)

Syntagmatic structure: a Finite Machine Model



Revisions needed to this model:

1. Reset (nternal repetition)
2. Insertion of parenthetics
3. Variables for declination
4. Interpolation of unstressed syllables
5. Constraints on accent sequences

Pierrehumbert (1980)

The finite depth grammar of the Prosodic Hierarchy

Prosodic Category inventory:

PC = {Utt, IP, PhP, PrWd, omega, Ft phi, syll, mora, segment}

Prosodic Hierarchy ordering:

L = <Utt, IP, PhP, PrWd, omega, Ft phi, syll, mora, segment>

l1 = Utt, l2 = IP, … l9 = segment

Structural constraints on Pros

But iterative recursion at the same rank is ok.

Strict Layering Hypothesis:

PC at L_i dominates only PC L_{i+1}!

- Fixed depth (no recursion): No PC at L_i dominates a PC at L_{i+1}
- Exhaustivity: All PCs at L_i are dominated by a single PC at L_{i-1}

Headedness:

- Every PC at L_i immediately dominates a PC at L_{i+1}

Prosodic grammar – tone sandhi

Downstep, upstep in Niger-Congo tone systems

Tem (ISO 639-3 *kth*) as a clear case example:

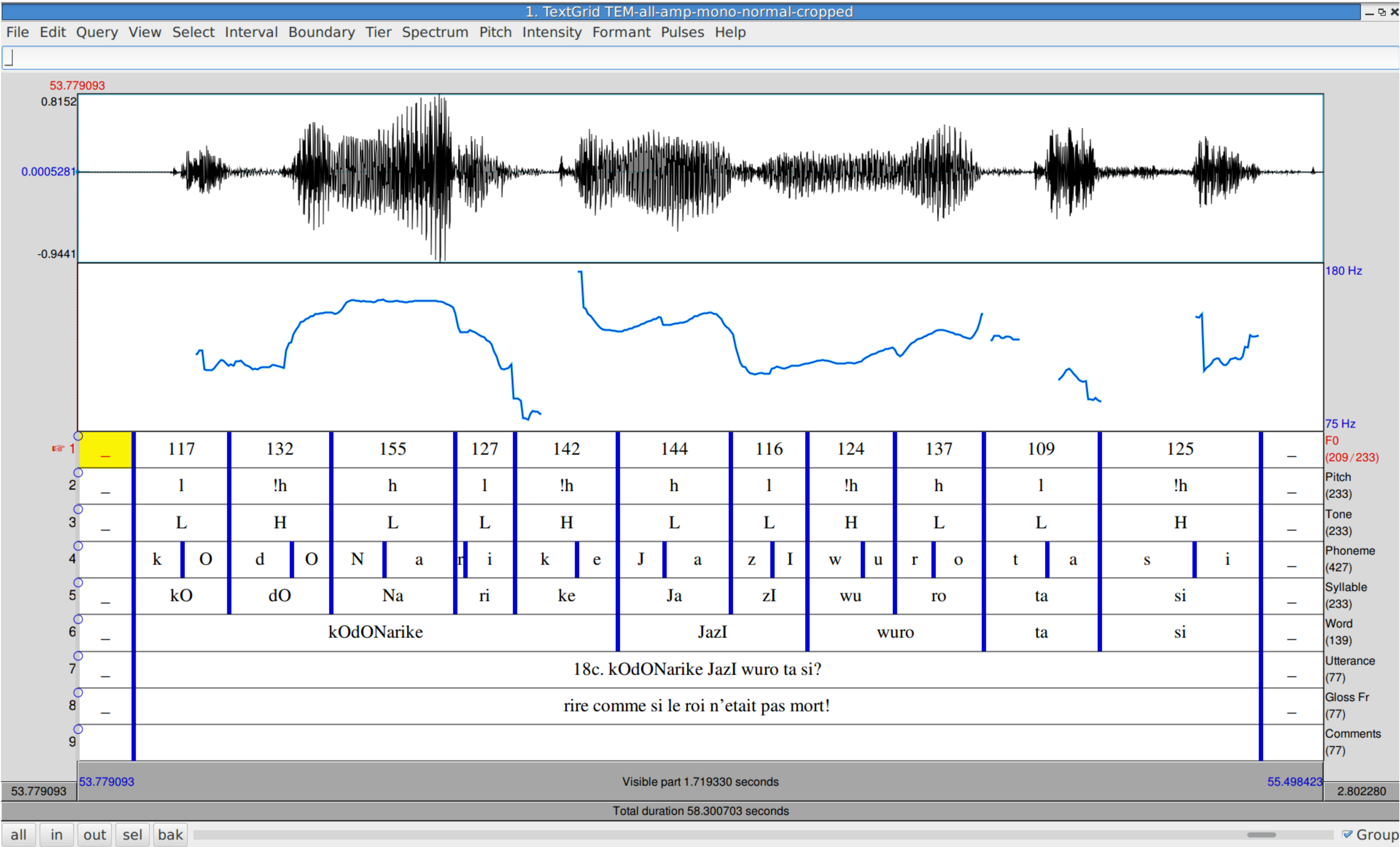
- Phonetic interpretation of Tem tone sequences:
  - inventory of 2 tones, H and L
  - L H: partial automatic downstep producing terracing
  - H L: complete automatic upstep
  - L semiterrace sequences: quasi-constant low
  - Initial H, L: extra high, extra low, respectively
    - Notation:
      - Underlying *tone categories*: upper case (H, L)
      - Surface *phonetic pitch categories*: lower case (h, !h, l, ^l)

Thus, in a traditional notation:

H → !h / L __ (terrace restart by automatic partial downstep)

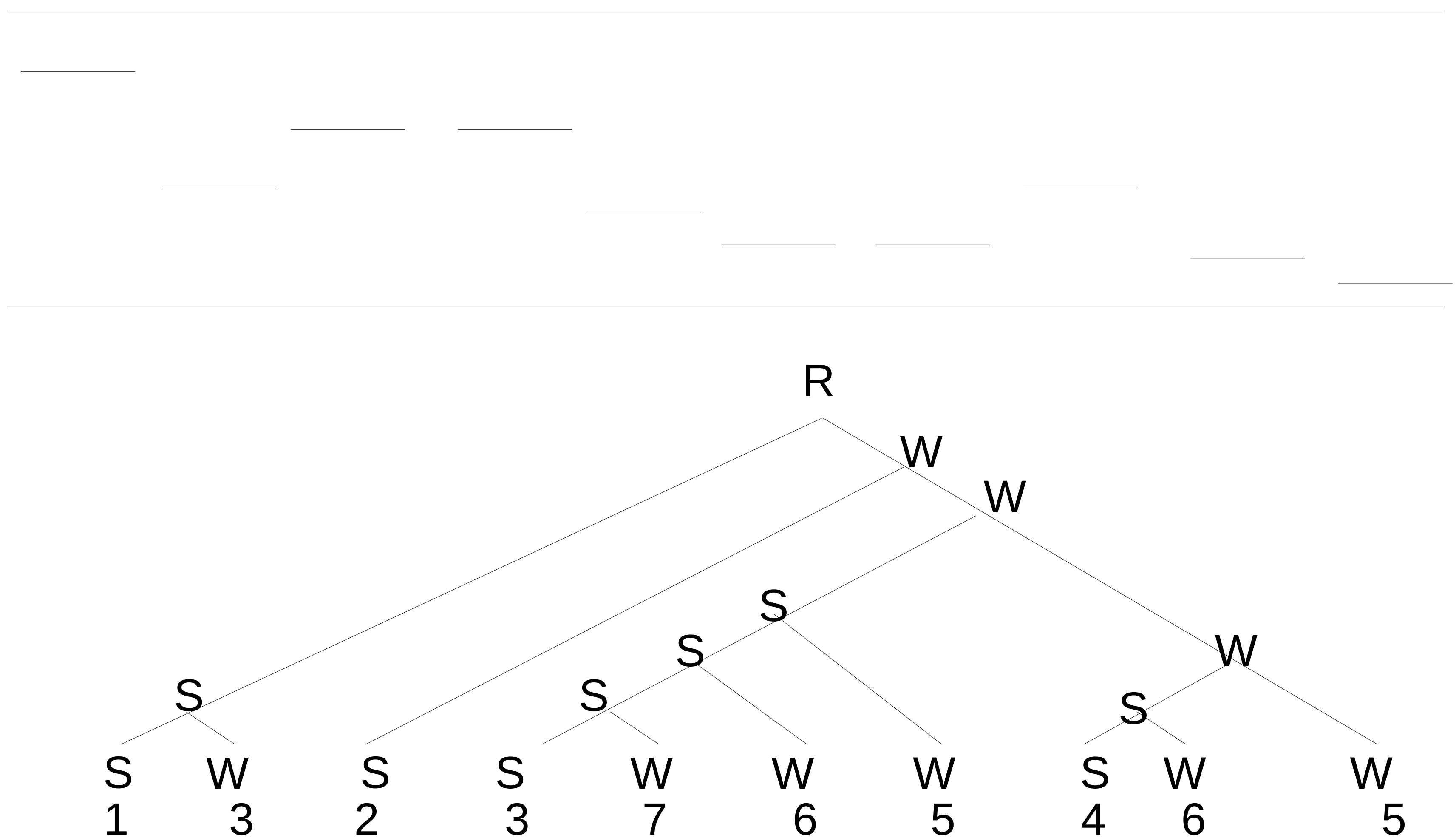
L → ^l / H __ (semiterrace extension by automatic total upstep)

Niger-Congo terraced tone systems



TEM kodoNa

Niger-Congo terraced tone systems

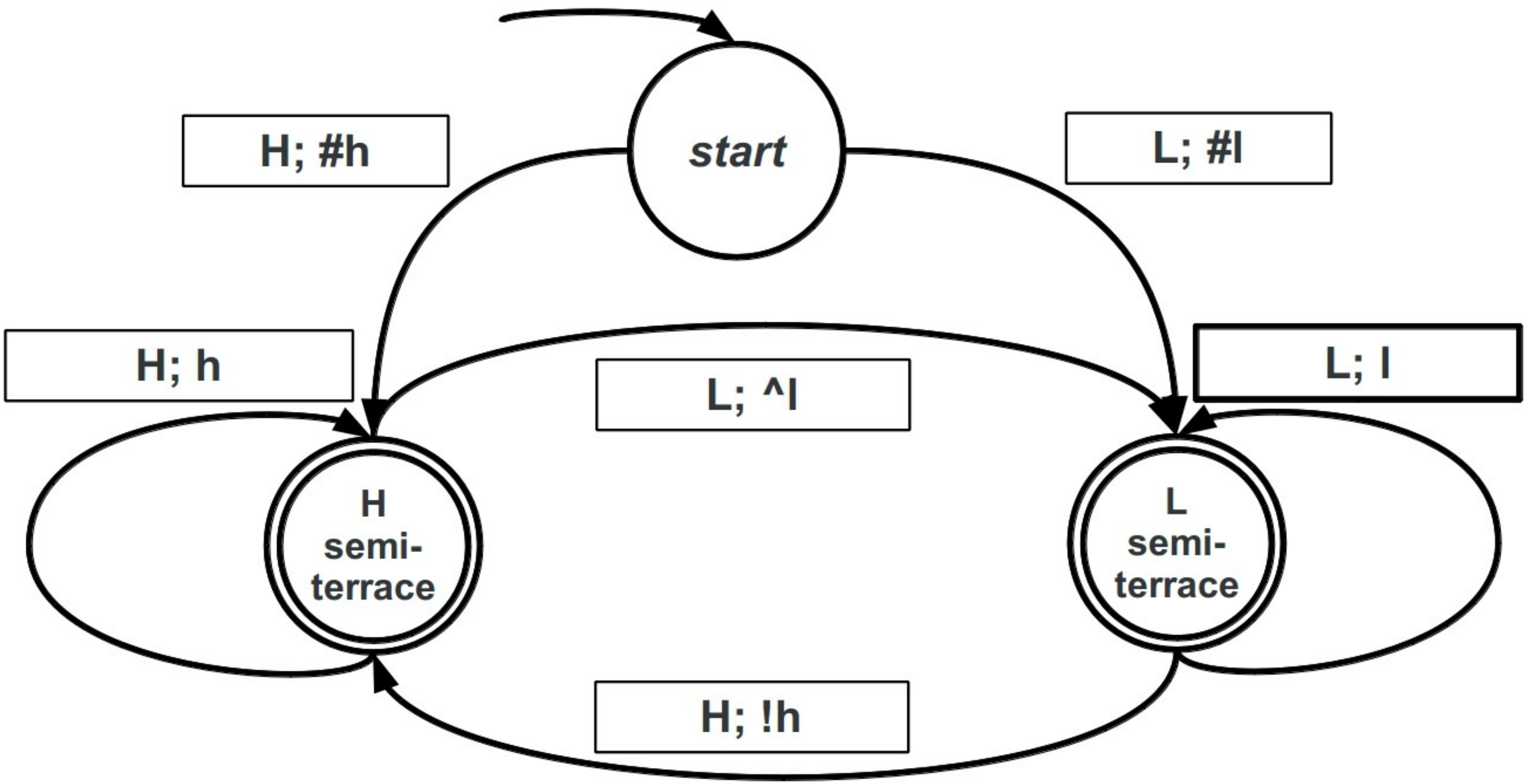


H	L	H	H	L	L	L	H	L	L
h	!l	^h	h	!l	l	l	!h	^l	l

Niger-Congo terraced tone systems

H	L	H	H	L	L	L	H	L	L
h	!l	^h	h	!l	l	l	!h	^l	l

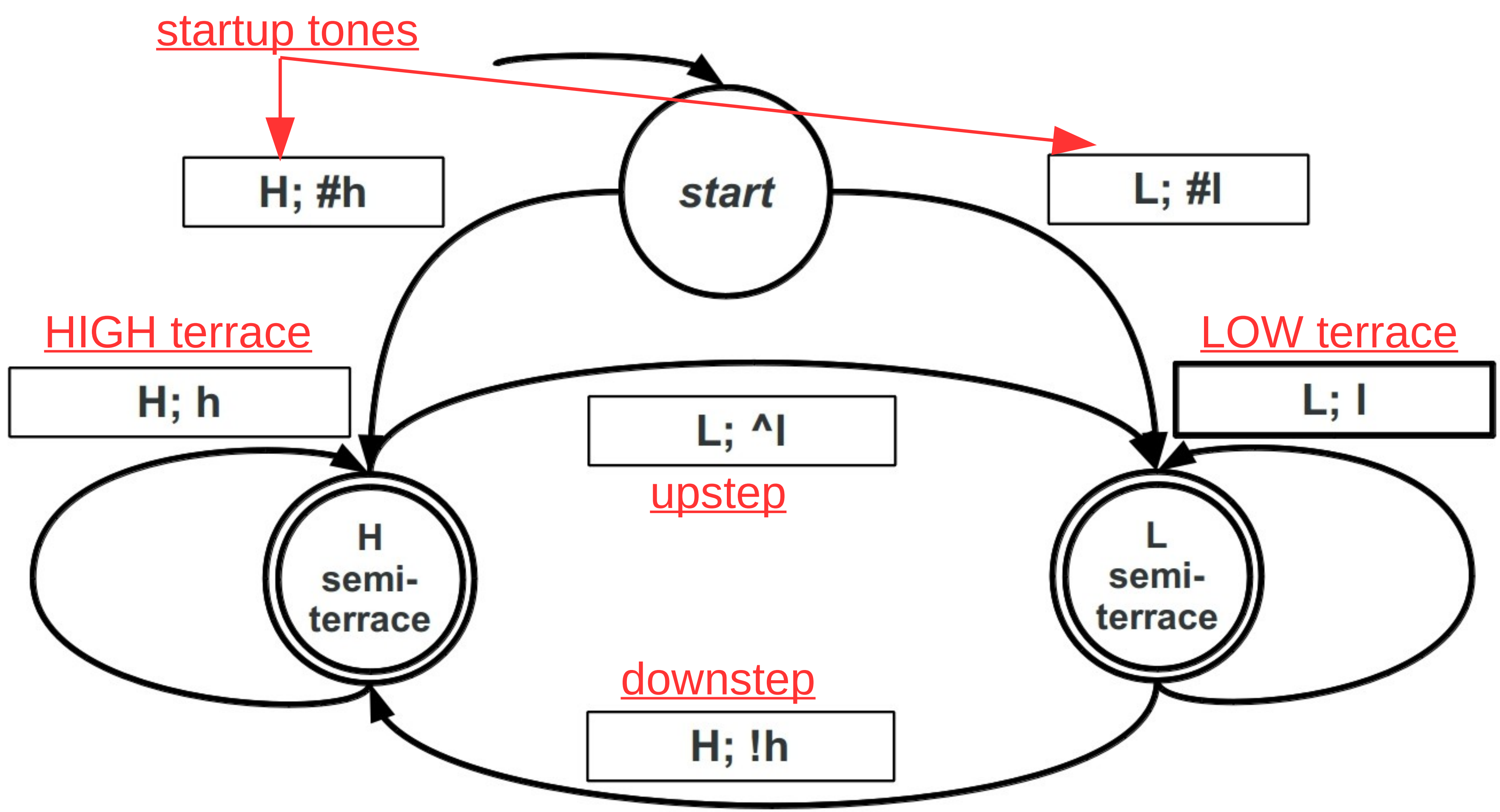
Niger-Congo terraced tone systems



Relevant contexts for tones
start and *end*
H and *L* terrace cycles
HL and *LH* terrace transitions

The graph defines 6 contexts (edges) for tone-allotone (tone-pitch) relations.

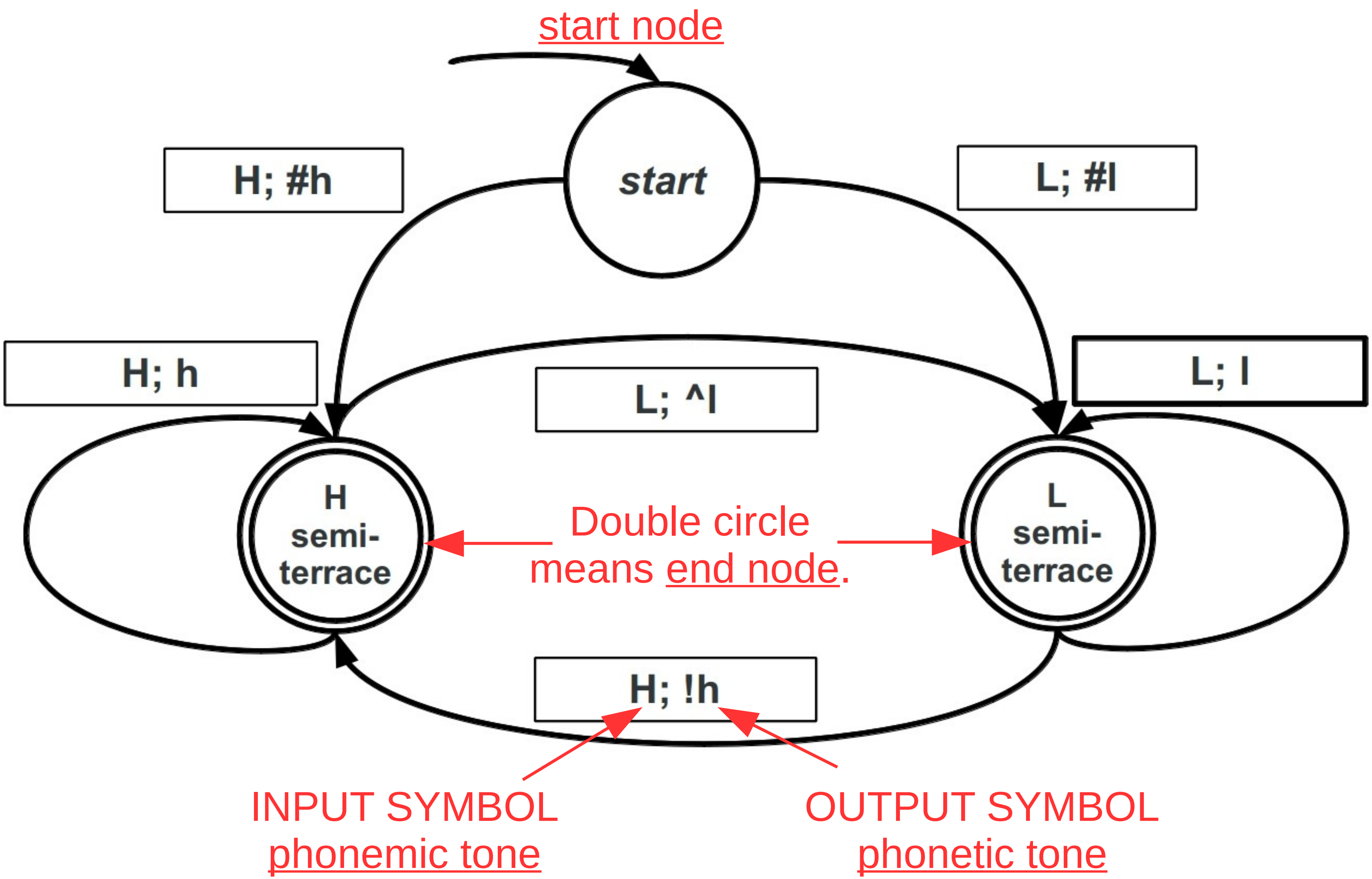
Niger-Congo terraced tone systems



Allotone pitch modification
#h, *#l* and *end*
h and *l* terrace cycles
^l and *!h* terrace transitions

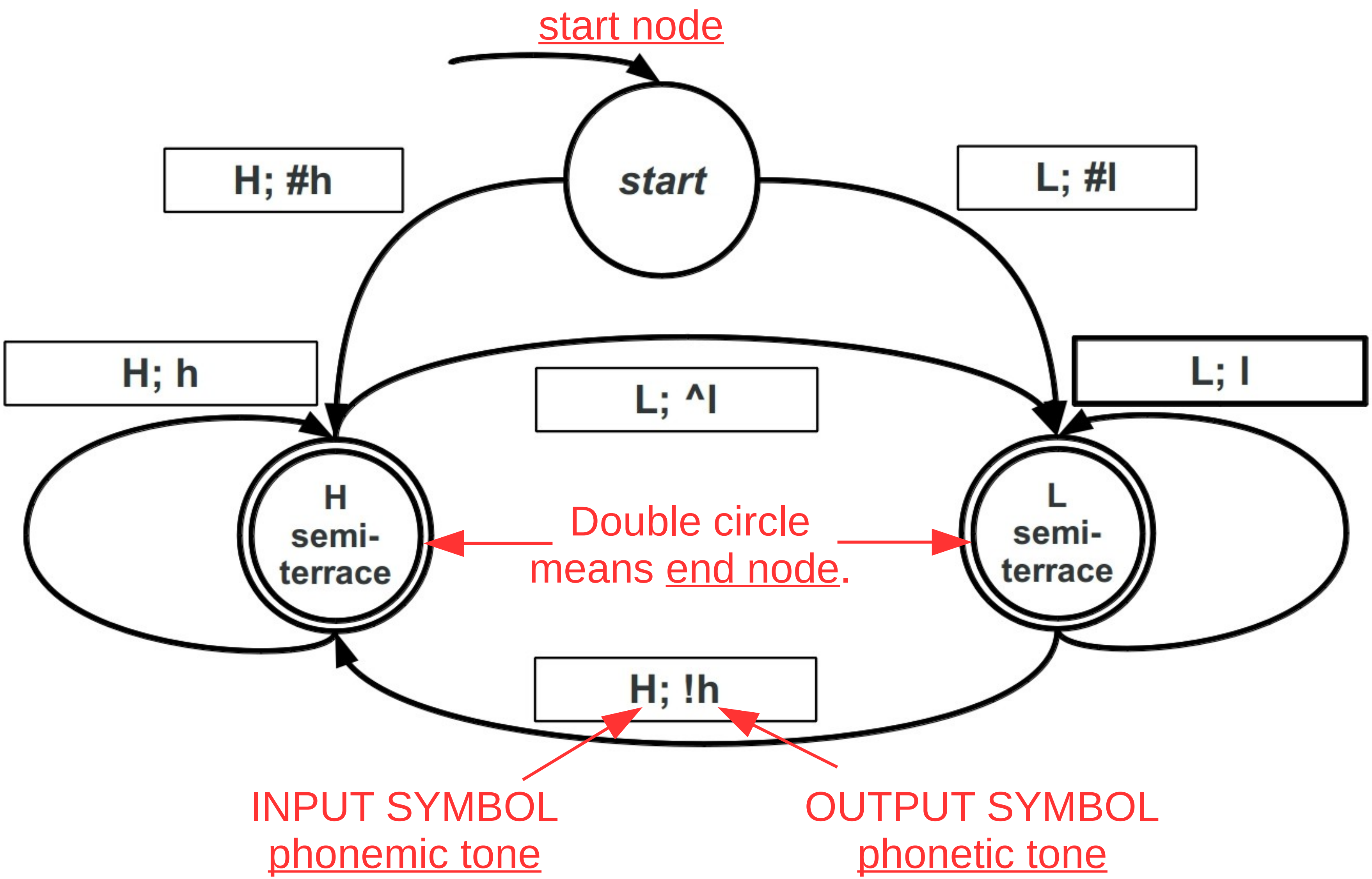
In addition to start, terrace and transition allotones, end allotones also need to be made explicit.

So how does this work?



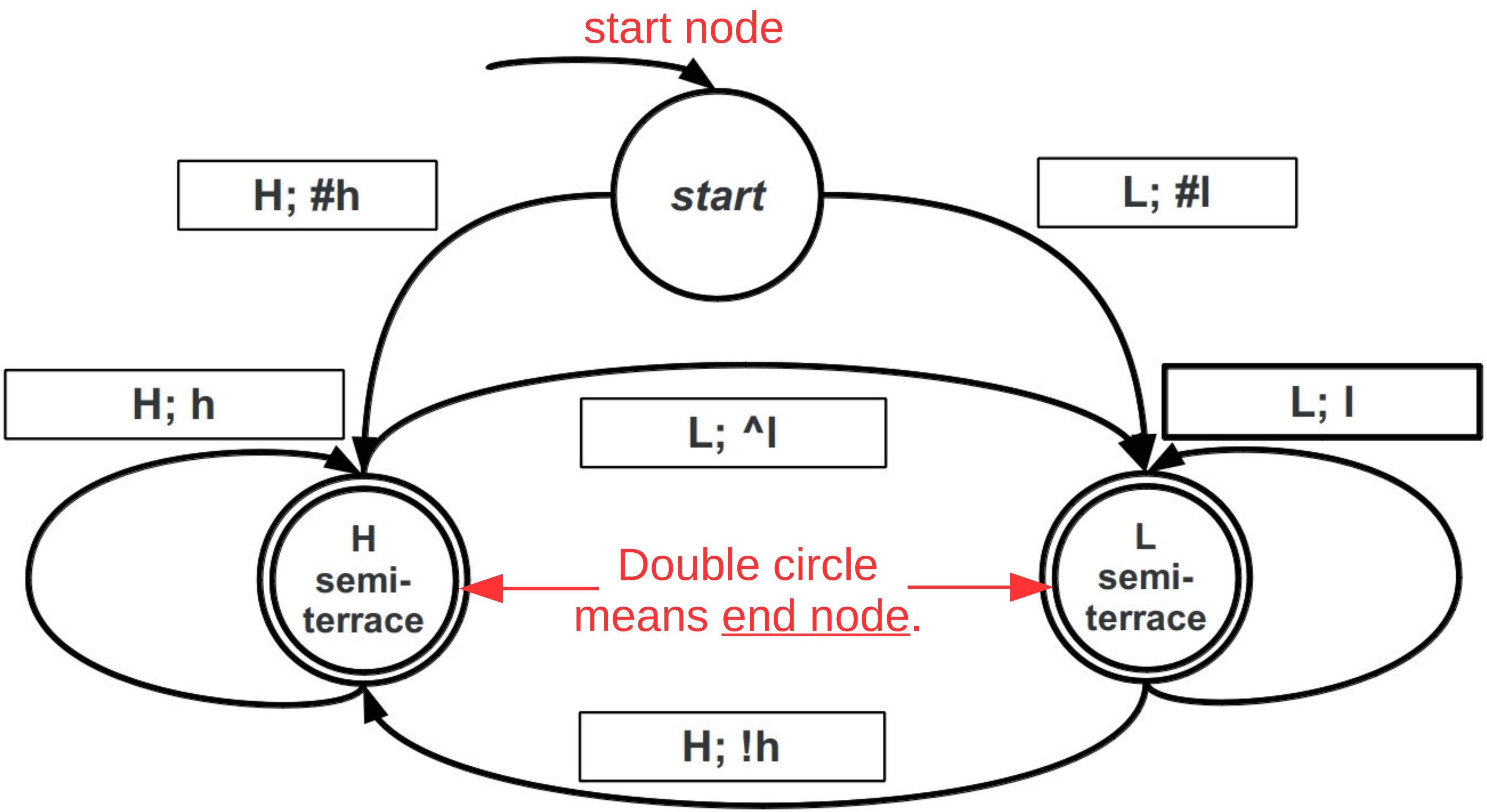
H H L L H L H H H → #h h ^l l !h ^l !h h h

So how does this work? Your turn!



L H H L L H → ? ? ? ? ? ?

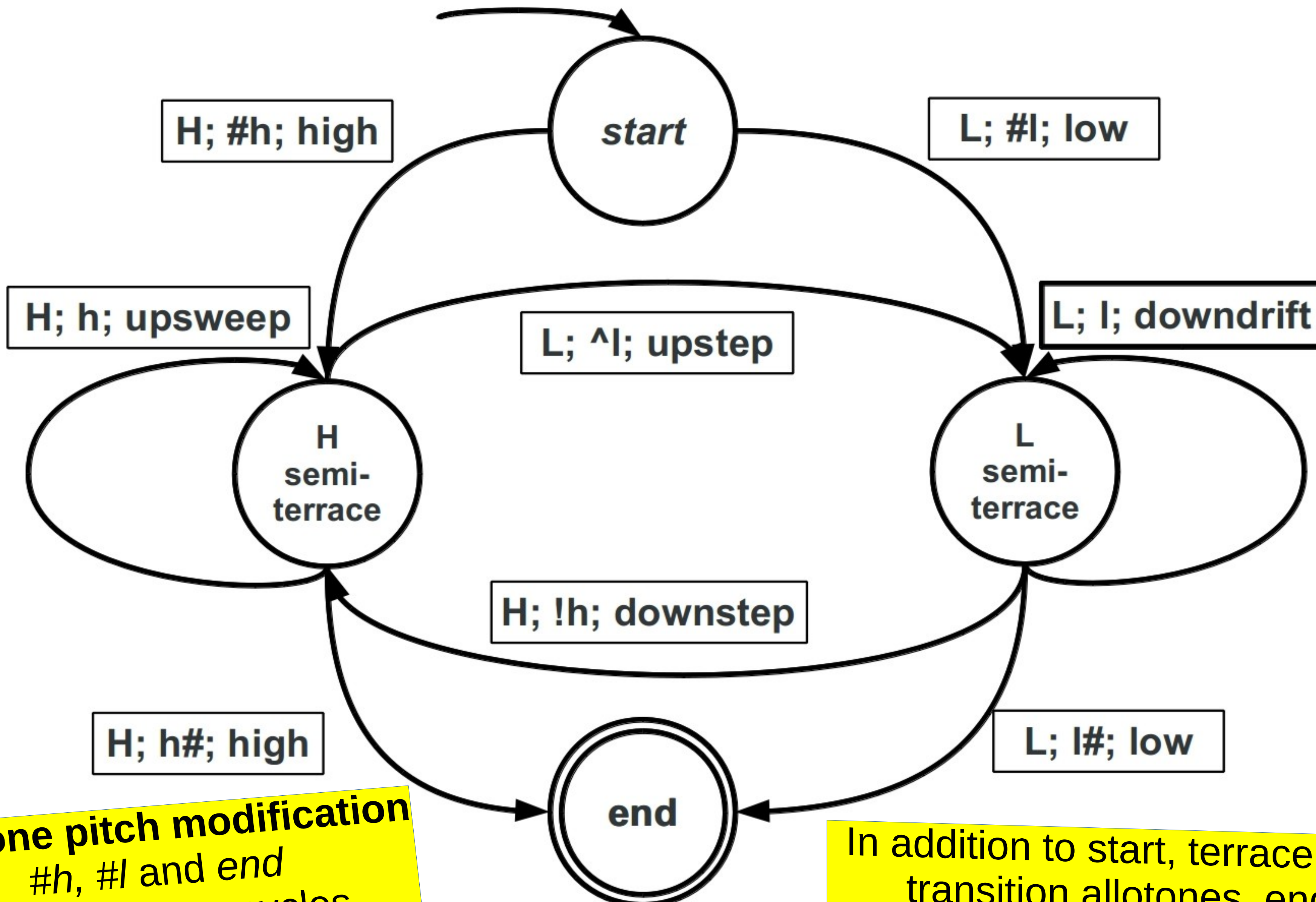
So how does this work?



1. Start at the start node with an input string of tones and an empty output string.
2. Choose an arrow with a left-hand symbol which matches the next input tone.
   1. Add the right-hand tone to your output string.
   2. Continue to the next input tone and the node at the end of the arrow.
3. When the last input tone has been successfully dealt with in this way, then if you are at an end node you have finished.

- Otherwise the model is rubbish and you need to revise it! :)

Downstep, upstep in Niger-Congo tone systems

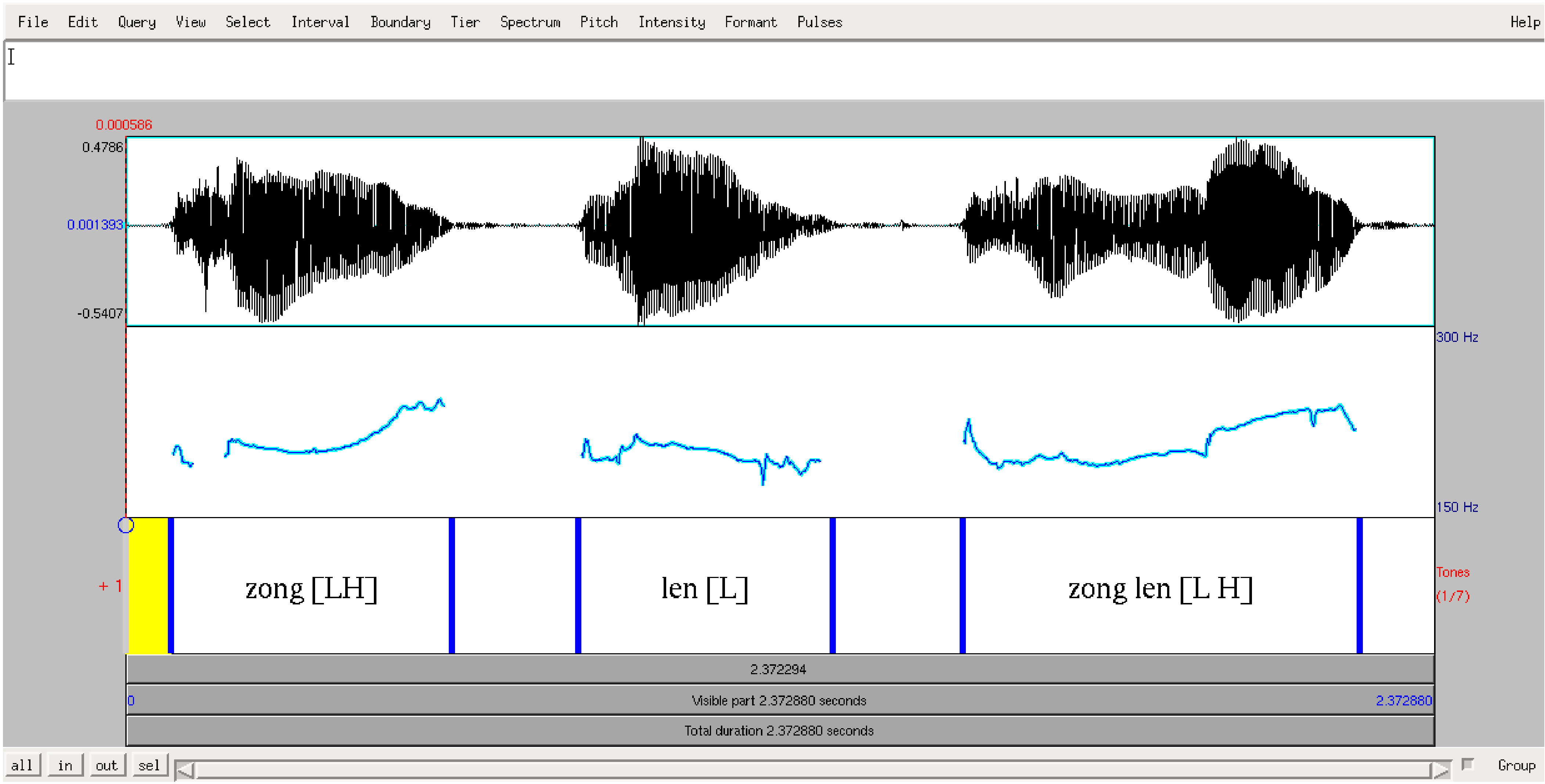


Allotone pitch modification
#h, *#l* and *end*
h and *l* terrace cycles
^l and *!h* terrace transitions

In addition to start, terrace and transition allotones, end allotones also need to be made explicit.

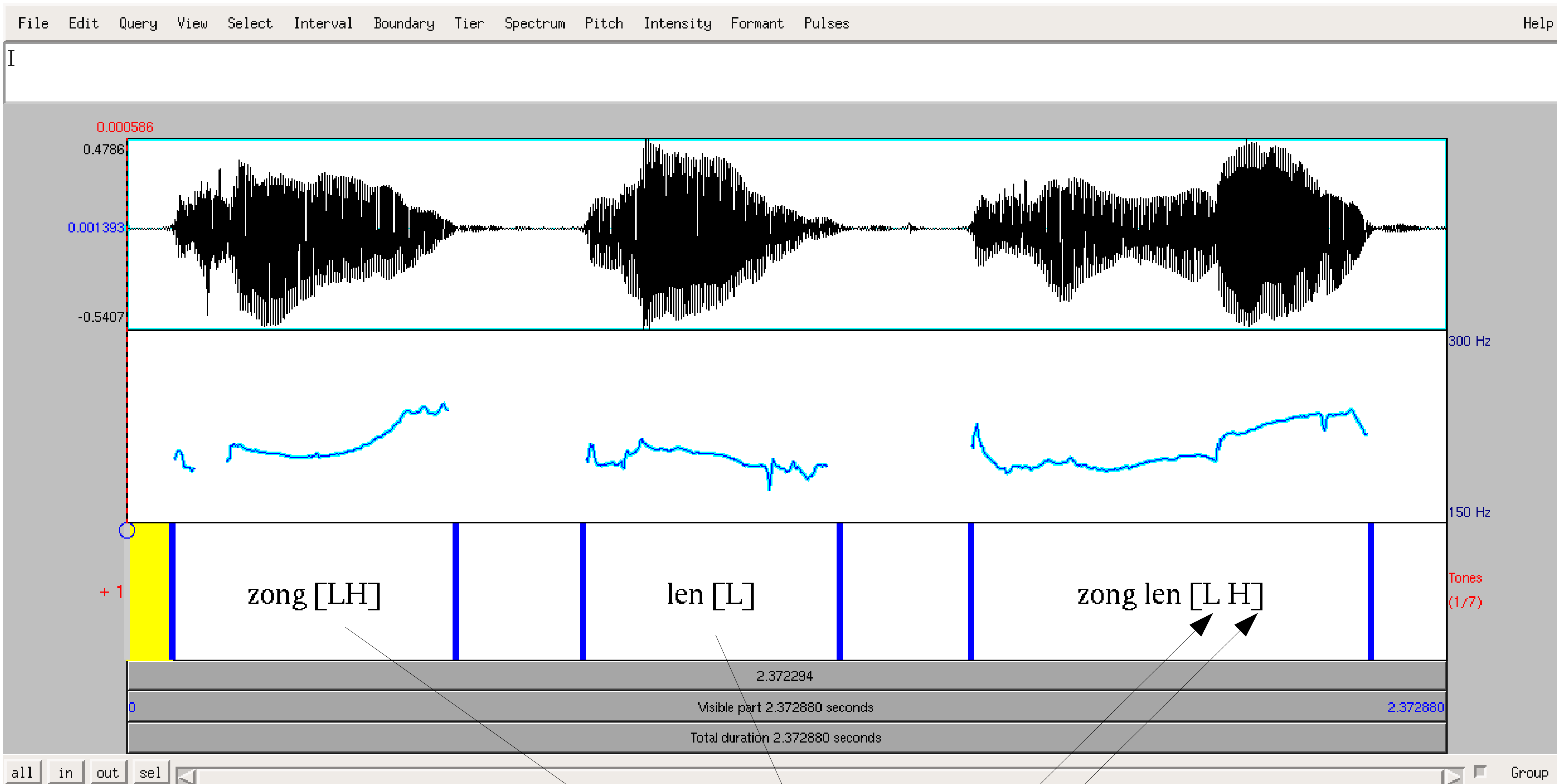
Sino-Tibetan tone Kuki-Thadou

Kuki-Thadou



"monkey big"

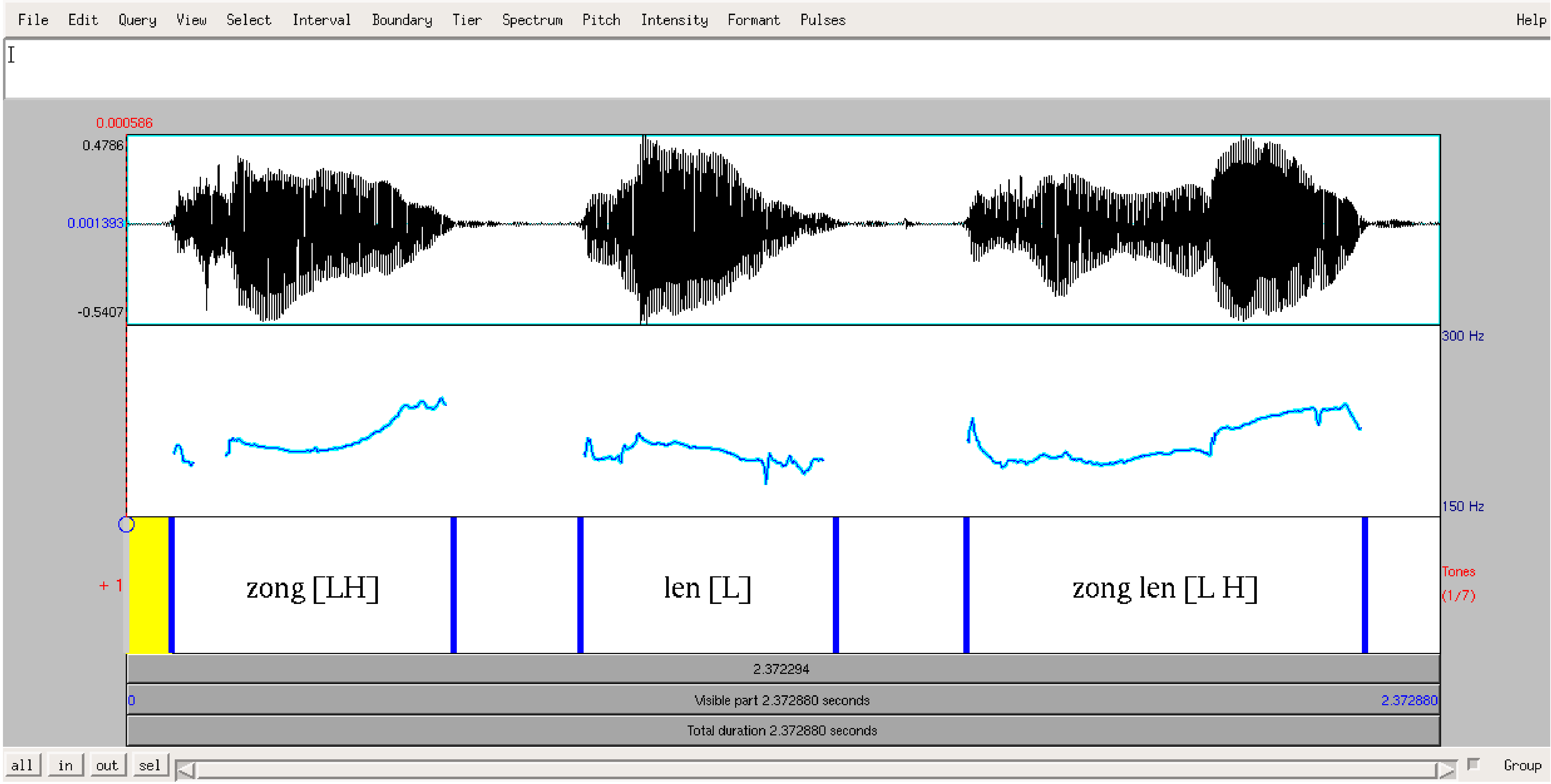
Kuki-Thadou



zong len zonglen

Linear tone sandhi rule: L H + L → L H

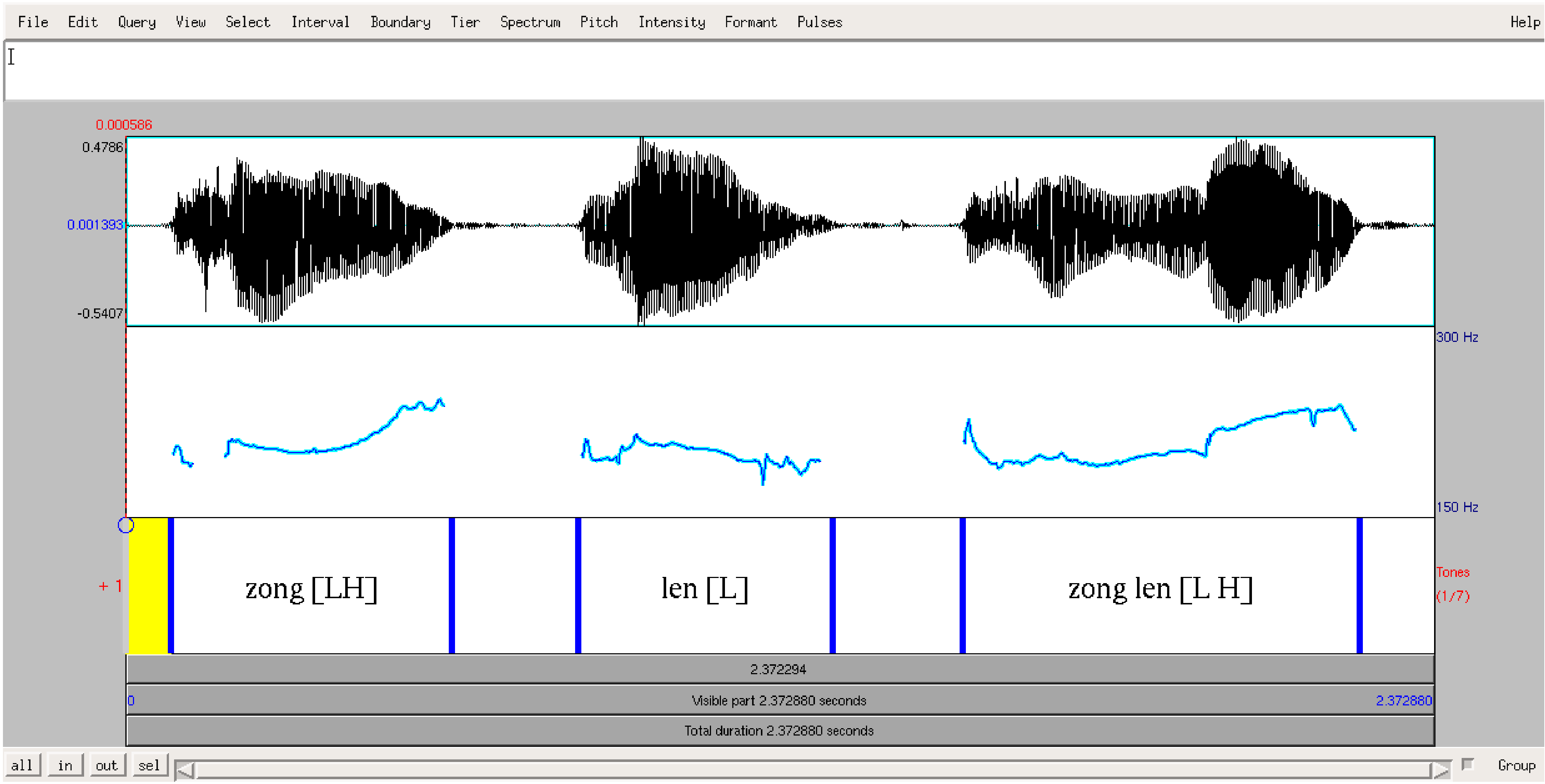
Kuki-Thadou

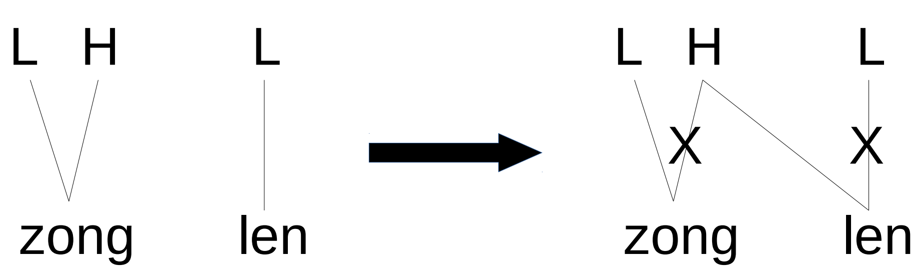


L → H / H + _ H → L / _ + L	L H + L L H + H L H + H	L → H / H + _ H → L / _ + L	L H + L L H + L L H + L

These classic tone sandhi rules: check different rule orderings – the rules are not ordered, but apply simultaneously.

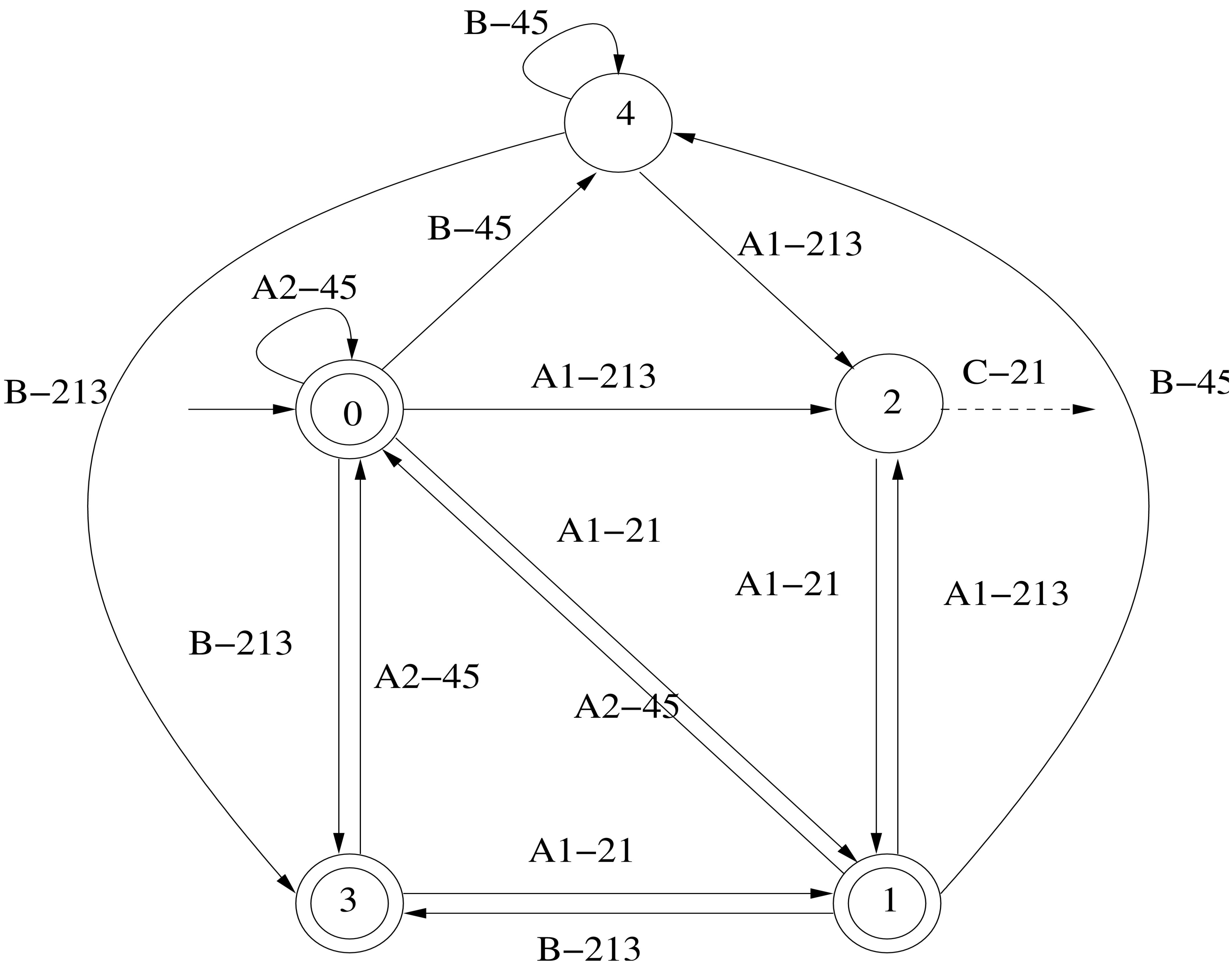
Kuki-Thadou


File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
0.000586
0.4786
0.001393
-0.5407
300 Hz
150 Hz
+ 1
zong [LH]
len [L]
zong len [L H]
Tones
(1/7)
2.372294
0
Visible part 2.372880 seconds
2.372880
Total duration 2.372880 seconds
all in out sel
Group


L H L
zong len
L H L
X X
zong len

Sino-Tibetan tone: Tianjin Mandarin

Tone sandhi in Chinese tonal systems: Tianjin Mandarin



Jansche, M. 1998. A Two-level Take on Tianjin Tone. In: I. Kruij-Korbayova, ed. *Proceedings of the Third ESSLLI Student Session*. Chapter 12.

Tone sandhi in Chinese tonal systems: Tianjin Mandarin

- Sorry, can’t give you any more information than this, but I can let you have the article by Martin Jansche :)

Summary: what you should know about by now

- Prosodic grammar:
  - different notations for transcribing and visualising prosody
  - different models for representing the structure of prosody
  - different patterns for typologically different languages