Mende Tone Patterns Revisited: Tone Mapping as Local Constraint Evaluation

Jochen Trommer
Universität Potsdam
Institut für Linguistik/Allgemeine Sprachwissenschaft
Postfach 601553 14415 Potsdam, Germany
trommer@ling.uni-potsdam.de
URL: http://www.ling.uni-potsdam.de/gk/stipendiaten/trommer/

Abstract

In this paper I present two constraint-based finite-state accounts for the classical problem of the tone patterns in Mende noun roots (Leben, 1978). The first model is inspired by "Declarative Phonology" (Bird et al., 1992) in using exclusively surface-true, inviolable constraints. The mapping of tones to segments is implemented by simple operations on automata. The restrictive effects of the auto-segmental mapping algorithm (Goldsmith, 1976) are captured by a single constraint. In the second model, as in Optimality Theory (OT; Prince and Smolensky, 1993), unviolability of constraints is abandoned. The effects of tone mapping are derived from well-documented constraints on phonological markedness and an evaluation algorithm for automata that - contrary to formalizations of standard OT (Ellison, 1994) - works completely locally. This approach is shown to be conceptually superior to the monotonic account as well as to standard OT-accounts (e.g. Heiberg, 1999) that rely heavily on constraints of the Generalized-Alignment type.

1 The Data and their Standard Analysis

The data from Mende in (1) (Leben, 1978), though not unproblematic empirically (cf. Odden, 1995, pg. 461) have always been one of the most used illustrations if not justifications for the auto-segmental treatment of tone.

The central problem posed by the data is to account for the heavily restricted set of tone patterns given the 5 surfacing Mende tones: High (**H**, *kɔ̈*), Low, (**L**,

[1]The notation I choose for tones differs from the traditional one. *a* corresponds to traditional *à*, *ä* to *á*, *á* to *ǎ*, *à* to *â* and *â* to *ã*.

(1) Tone Patterns in Mende Noun Stems [1]

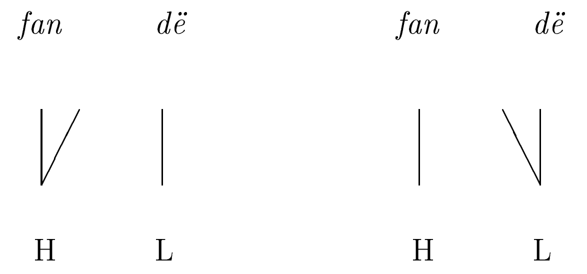
H:	*kɔ̈*	"war"	*pɛ̈lɛ̈*	"house"	*häwämä*	"waistline"
L:	*kpa*	"debt"	*bɛlɛ*	"trousers"	*kpakali*	"tripod chair"
HL:	*mbù*	"owl"	*ngïla*	"dog"	*fëlama*	"junction"
LH:	*mbá*	"rice"	*fandë*	"cotton"	*ndavülä*	"sling"
LHL:	*mbâ*	"companion"	*nyahá*	"woman"	*nikïli*	"groundnut"

(Leben, 1978, pg. 186)

kpa), Fall (**F**,*mbù*), Rise, (**R**, *mbá*) and Rise-Fall (**T**[2] ,*mbâ*). For example in three-syllable stems we would expect 5 x 5 x 5 = 125 different possibilities to distribute the tones, but there are no more patterns than tones, namely 5 . In his influential thesis Goldsmith (1976) proposes, that despite the surface apparency there are in fact only five patterns for *all* stems , namely: **H, L, HL, LH, LHL**. These tone patterns in his model are represented separately from the segmental string on the so-called tone-tier and then associated with segments. Contour tones thus following an older tradition in African Linguistics are not conceived as atomic tones, but as single segments linked to multiple tone units and long vowels as one single tone linked to two segments. As can be seen in (3), under this analysis two- and three-syllable-stems differ from the corresponding one-syllable stem only with respect to linking.

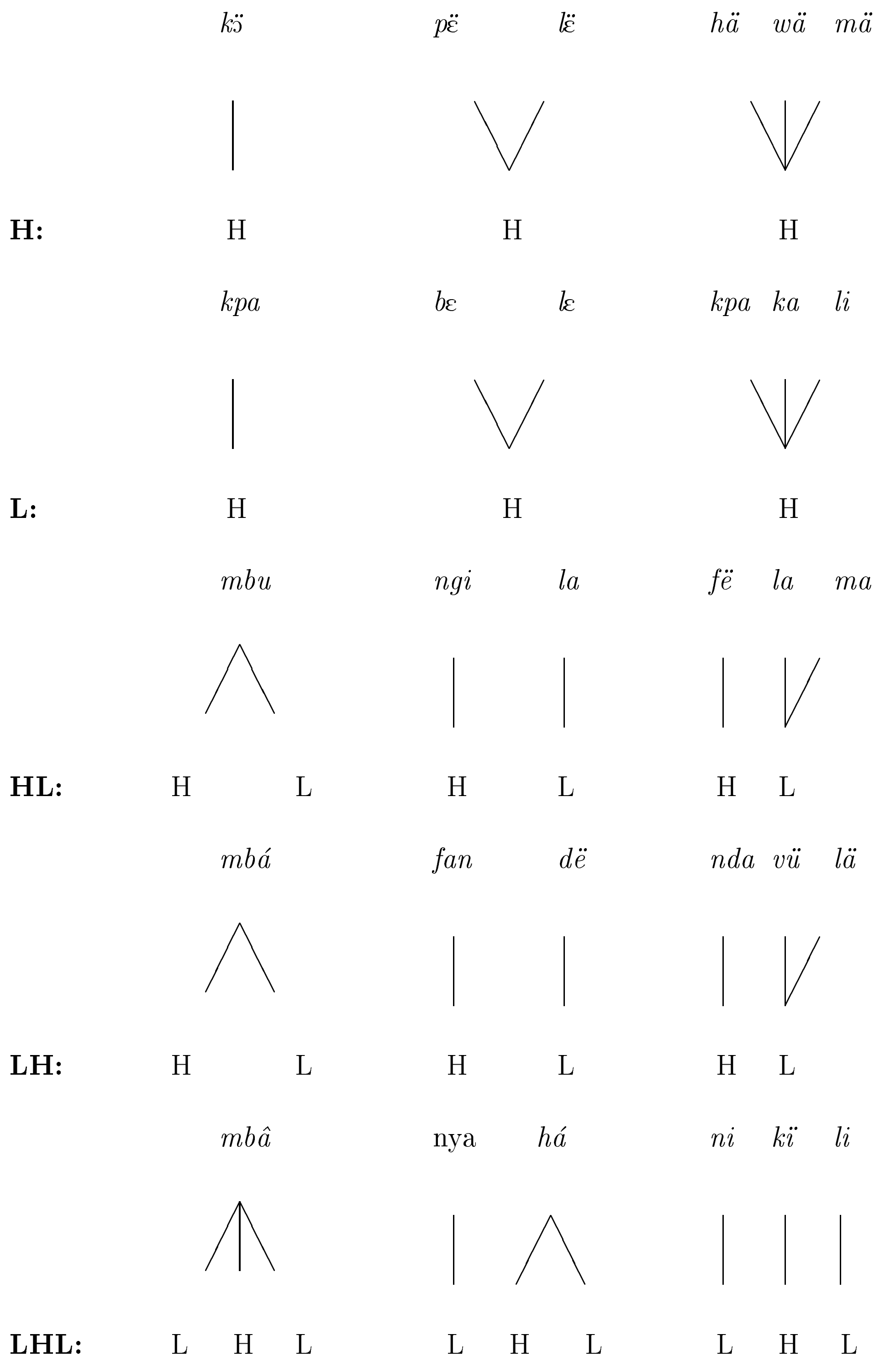
However, a given tonal pattern doesn't lead a priori to exactly one surface tone representation. For example for *fandë* the following realizations would also be possible:

(2) Possible Associations for fandë

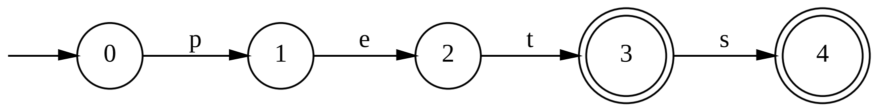


[2]**T** is used to get an unambiguous single letter abbreviation for Rise-Fall. It refers to the fact that this is the only contour corresponding to **t**-hree single tones.

(3) Auto-Segmental Representation of the Mende Data



(6) Simple Finite-State Automaton



Goldsmith ensures the correct mapping through the following algorithm:

(4) Tone Mapping:
  a. Associate the first tone with the first syllable, the second tone with the second syllable, and so on, until all tones or syllables are exhausted.
  b. Tones or syllables, not associated as a result of (a) are subject to the well-formedness condition

Tones and segments not associated with an element of the other tier must be associated, according to a special well-formedness condition(WFC):

(5) Well-Formedness Condition (WFC)
  a. Every tone is associated with some syllable
  b. Every syllable is associated with some tone.
  c. Association lines may not cross.

2 Finite-State Models of Phonological and Morphological Description

Finite-state machines are the most restrictive subfamily from the family of formal languages known as Chomsky hierarchy (cf. Hopcroft and Ullman, 1969). Since the seminating work of Kaplan and Kay (1998) refinding results of Johnson (1972), and Koskenniemi (1983) finite-state devices play a central role in attempts to formalize phonological concepts like rewrite rules (Kaplan and Kay, 1998), auto-segmental phonology (Bird and Ellison, 1994) and Optimality Theory (Ellison, 1994). The simplest finite-state machines are finite-state automata.

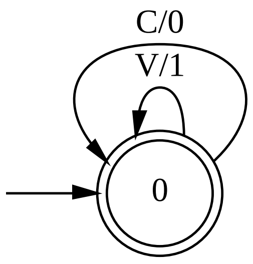
A finite-state automaton (FSA) consists informally speaking of transitions over symbols between a finite set of states some of which are final and/or initial.[3] In diagrams transitions are depicted as labeled arrows. Final states are marked by a double circle, (3 and 4 in (6)) initial states by a single arrow pointing to them (0):

A string $B_1, B_2, \ldots, B_n$ is *accepted* or *generated* by a FSA, iff there is a start state S_A, a final state S_E and transitions $S_A - B_1 \rightarrow S_1, S_1 - B_2 \rightarrow$

[3]For more formal definitions see Hopcroft and Ullman (1969)

$S_2, \ldots, S_{n-1} - B_n \rightarrow S_E$. Thus the string "pet" is accepted by the automaton in (6) since you can go from 1 (start state) to 4 (final state) reading "pet". A simple but nontrivial extension of FSAs are Finite-State Transducers (FSTs).The only difference is that in FSTs transitions don't go over simple symbols but over ordered pairs of symbols, thus mapping strings onto strings. For example the following transducer maps strings consisting of Cs and Vs into strings of 0 and 1 replacing each C by 0 and each V by 1

(7) Simple Finite-State Transducer



Note that mapping here isn't meant procedurally, since You can use the transducer equally well the other way around for mapping 0s and 1s to Cs and Vs a .

FSAs are generatively equivalent to regular expressions (REs). REs can be conceived as abbreviation devices for string sets. Thus $C(C)?V^*C^+$ means one C followed optionally by another C followed by $0, 1, \ldots$ many Vs followed by $1, 2, \ldots$ many Cs. REs can be constructed recursively over a finite alphabet by the operations in (8).

Often it is more convenient to notate in REs, but to define algorithms over automata. Since there are standard procedures to create equivalent automata (expressions) for each expression (automaton) this is formally without problems.

(8) Syntax of Regular Expressions

Disjunction	$(R_1\|\ldots\|R_n)$	R_1 or ... or R_n
Concatenation	R_1R_2	The String R_1 and R_2
Kleene-closure	R^*	none or arbitrary many Rs in sequence
+-closure	R^+	one or arbitrary many Rs in sequence
Optionality	$R?$	one or none R
Definition[4]	$\{X\}$	the regular expression defined as X

One of the main advantages of FSAs is the possibility of intersecting them. If there are two automata you can create by algorithm a third FSA that accepts all

[4]Definitions are introduced by a line containing the definition followed by a regular expression. Thus the line "{X} $C(C)?V^*C^+$" defines X as the described RE.

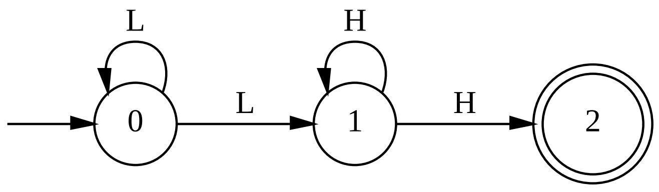
and only the strings, accepted by both automata. This allows a modular description of language data, since different constraints can be conceived as automata partially describing the facts, while it is guaranteed that their together is formally again a finite-state description.

3 Mapping Tone Patterns and Constraints to Finite-State Machines

3.1 Mapping Tone Patterns to Automata

Under a segmental view a tone pattern like **HLH** means one or more **H** tone vowels followed by one or more **L** tone vowels followed on its side by one or more **H** tone vowels. This can be easily expressed by the regular expression $H^+L^+H^+$. More generally each tone pattern of the form $T_1, T_2, \ldots, T_n$ can be implemented as a regular expression ${T_1}^+{T_2}^+ \ldots {T_n}^+$. There are two complications: contour tones and intervening consonants. For the algorithm integrating these two,we start by constructing the finite-state automaton that's equivalent to the obtained regular expression, e.g. for L^+H^+ (**LH**):

(9) Automaton for L^+H^+



Assuming that contour tones realize sequences of simple tones, we suppose for a language with contour tones a function F that maps each contour C onto a corresponding tone sequence, observing the OCP.[5]

For each such tone sequence $T_S = T_1, T_2, \ldots, T_n$ and each tone pattern automaton there is a finite set S of ordered state pairs (S_1, S_n), such that you can traverse the automaton from S_1 to S_n reading T_S . For each tone sequence T_S in the language and each such state pair we add the transition $S_1 - C \rightarrow S_n$ to the automaton where C is the contour tone corresponding to T_S.

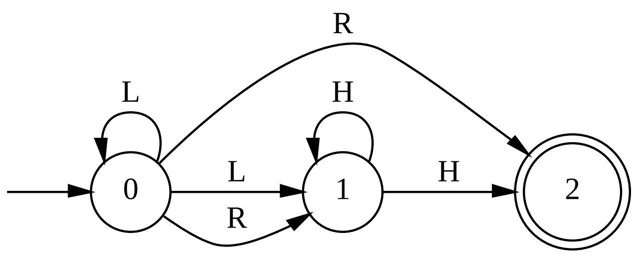
The correspondences between contours and Tone patterns in Mende are summarized in (10) ; included are the sets of state pairs for our example automaton. The modified automaton is depicted in (11)

[5]i.e. without adjacent identical tones like **HH**. Cf. Odden (1995)

(10) Mende Contour Tones and State Pairs for (9)

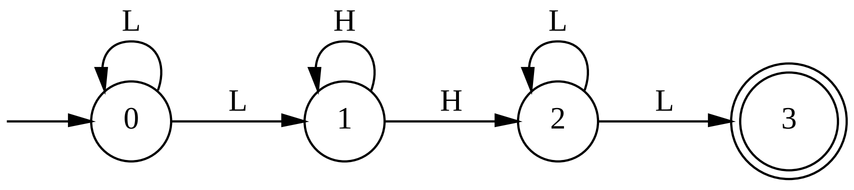
Contour	Tone Pattern	Set of State Pairs
R(ise)	**LH**	{(0,1),(0,2)}
F(all)	**HL**	{}
T(RiseFall)	**LHL**	{}

(11) Automaton for L^+H^+ with Contour Tones



The algorithm applies in a mirror fashion to the **HL** pattern and the **F** contour. Since there are no contours corresponding to tone patterns of length 1 it applies vacuously to the patterns H^+ and L^+. Thus for completeness I show the application to $L^+H^+L^+$

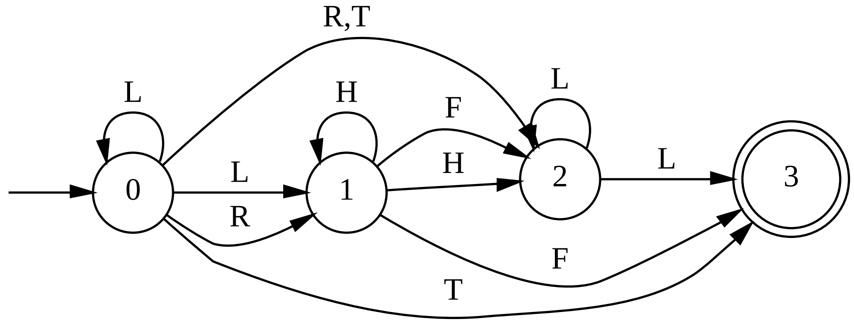
(12) Automaton for $L^+H^+L^+$ (without Contour Tones)



(13) State Pairs for (12)

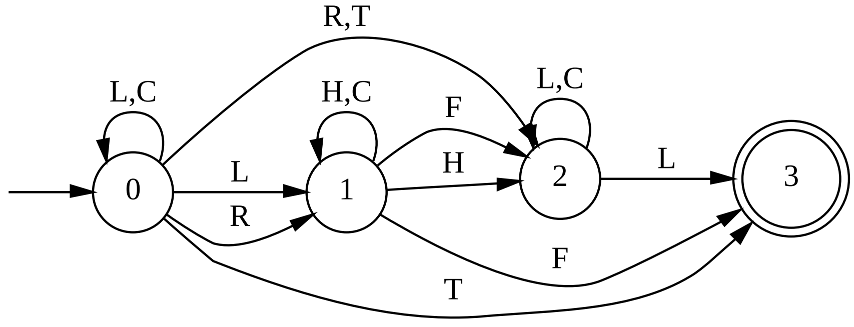
Contour	Tone Pattern	Set of State Pairs
R(ise)	**LH**	{(0,1),(0,2)}
F(all)	**HL**	{(1,2),(1,3)}
T(RiseFall)	**LHL**	{(0,2),(0,3)}

(14) Automaton for $L^+H^+L^+$ with Contour Tones



We have still neglected intervening consonants. They can be simply handled by adding C-arcs from each state to itself, so that arbitrarily many consonants can intervene, in the case of (14) this gives (15).

(15) Automaton for $L^+H^+L^+$ with Contour Tones and Consonants



Since the symbols on the arcs are still cover symbols like **H** and **L** they have to be replaced in a trivial way by arcs for the denoted segments, e.g. replacing each **H**-arc by arcs for ä, ï, ë, ö, ɔ̈, ɛ̈ . Here is the complete algorithm:

3.2 Applying the Algorithm to Phonological Constraints

The method used here can also be used to get contextual tone effects. Since in the Mende data the integration of contours in this case isn't strictly necessary I will illustrate the point with some data from Hausa (Newman, 1995, pg. 764).

Algorithm 1 Tone Mapping(P, F, C, M)

```
1: Input:
2: $P = P_1, \ldots, P_i$, a string of tones
3: $F_{(S \times T)}$, a function from strings of tones into contour tones
4: $S_C$, a set of consonants,
5: $F_{(C \times P)}$, a function from cover symbols into sets of phonemes
6: Output: $A$, a finite-state automaton
7:
8: $A \Leftarrow$ FSA($P_1{}^+, \ldots, P_i{}^+$)
9: for all strings $W$ in Domain($F_{(S \times T)}$) do
10:   for all pairs of states $(S_1, S_2)$ in $A$ do
11:     if there is a path over $W$ from $S_1$ to $S_2$ in $A$ then
12:       Add the transition $S_1 - F_{(S \times T)}(W) \rightarrow S_2$ to $A$
13:     end if
14:   end for
15: end for
16:
17: for all states $S_n$ in $A$ do
18:   add the transition $S_n - C \rightarrow S_n$ to $A$
19: end for
20:
21: for all transitions $S_1 - X \rightarrow S_2$ in $A$ do
22:   delete $S_1 - X \rightarrow S_2$ from $A$
23:   for all phonemes $P \in F_{(C \times P)}(X)$ do
24:     add transition $S_1 - P \rightarrow S_2$ to $A$
25:   end for
26: end for
```

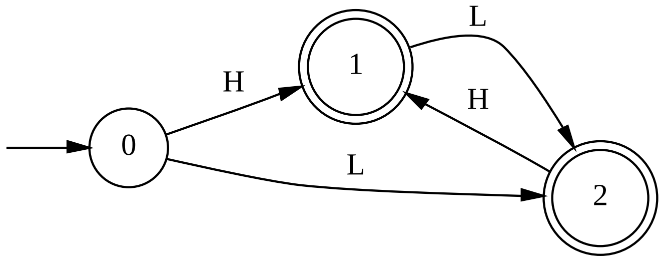
The Hausa "stabilizer" morpheme (*nee* "masculine or plural", *cee*, "feminine") has the tone polar to the preceding syllable, as shown in (16).

(16) Hausa Polar Tones

following **H**	following **L**	following **F** (**HL**)
riigää cee , "it's a gown"	*zööbee nëë*, "it's a ring"	*mài nëë*", it's oil"
keekë nee, "it's a bike"	*mööta cëë*, " it's a car"	*riigàr̃ cëë*, "it's the gown"

The fact that **F** is treated like a **L** tone here emerges naturally from the autosegmental analysis. In terms of automata it can be reconstructed, if we assume that phonological constraints too are mapped onto automata using the basic algorithm from 3.1 let's first assume the schematic candidate set $[HLF][HL]$, where the first disjunction stands for stems ending with a syllable of the respective tone, while $[HL]$ represents the (under-specified) stabilizer morpheme. I assume that polarity is an effect of under-specification conspirating with the Obligatory Contour Principle (OCP; Leben, 1978) demanding adjacent tones to be different.[6] A very strict form of the OCP can be implemented by the automaton in (18) which results by adding contours to (17).

(17) OCP without Contours



The only two tone sequences that are accepted by this automaton are **HL, LH, LF, FH** and **FF**. Intersecting these with the regular expression above we get **HL LH** and **FH**, exactly the correct results.

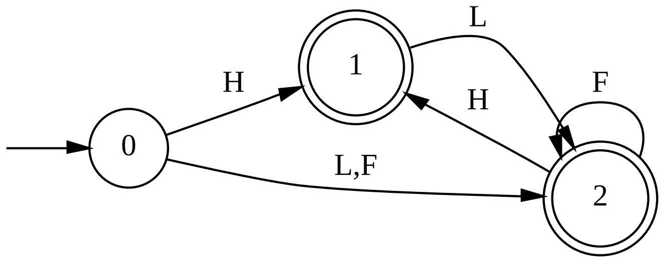
4 An Analysis with Unviolable Constraints

4.1 Some Shortcomings of Goldsmiths Account

There is a lot of stipulative in Goldsmiths account: First, let's note a problem, that results from the representational format Goldsmith chooses: A sequence of

[6]I.e. sequences like **HH** or **LL** are forbidden

(18) OCP with Contours Added



Ls and **Hs** is not understood as a partial description of the pertaining word. If it were the well-formedness condition were redundant, since e.g. **HL** would mean nothing else than a word (syllabic, segmental, etc.) manifesting a sequence of **Hs** followed by a sequence of **Ls** (possibly inside of contours) but nothing else[7]: In a finite-state framework we could assume e.g. for *fandë* a representation for the segmental content with under-specified tone values:

(19) f[aäáàâ]nd[eëéèê]

The tone pattern then would be (for the moment simplifying away the contours):

(20) $(C^{*}LC^{*})^{+}(C^{*}HC^{*})^{+}$,

where $C =$ [fndkplhwmbgvy], $H =$ [aeɛioɔu], $L =$ [äëɛ̈ïöɔ̈ü][8]

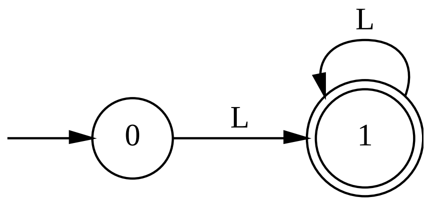
The only word that's described by both regular expressions is simply *fandë*. There is no need for conventions mediating them. Now under this approach at least for this relative simple example also the need for the tone mapping rule has vanished, since the correct result is achieved without it. This will not prove true for all cases but it hints at a second conceptual shortcoming in (4): The well-formedness-condition is valid also regarding the tones mapped by the (a)-part of the rule, but there it's achieved by a different mechanism. So at least part of (a.) seems to be doubling the WFC and thus to be dispensable.

4.2 The Candidate Sets Generated by Tone Pattern Constraints

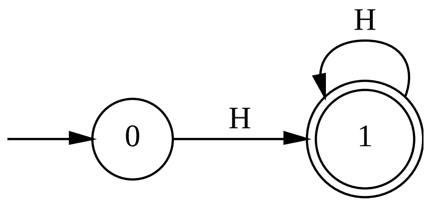
Again abstracting away from consonants and individual vowel quality individual stem representations are all of the types listed in (22).

[7]This is not an inherent problem of multi-tier-representations, but of their interpretation.
[8]Curly Brackets ("{}") inside of REs mark previously defined part expressions. Cf. (8).

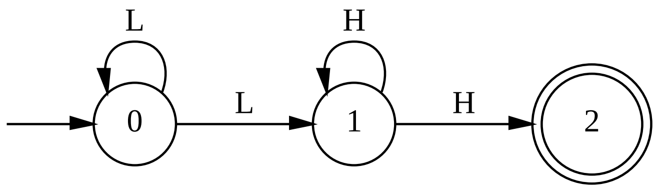
(21) Mende Tone Patterns as Automata



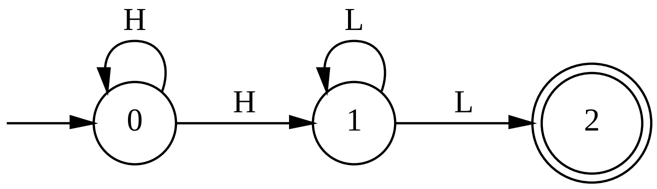
L



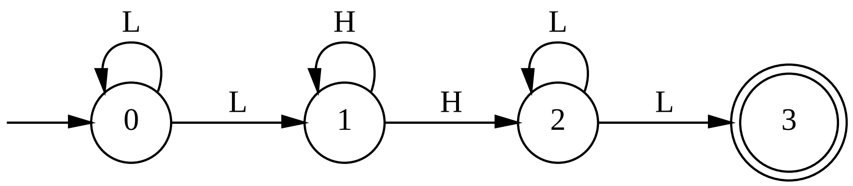
H



LH



HL



LHL

(22) Representations for Mende Stems

1 syllable	2 syllables	3 syllables
$[HLRFT]$	$[HLRFT][HLRFT]$	$[HLRFT][HLRFT][HLRFT]$

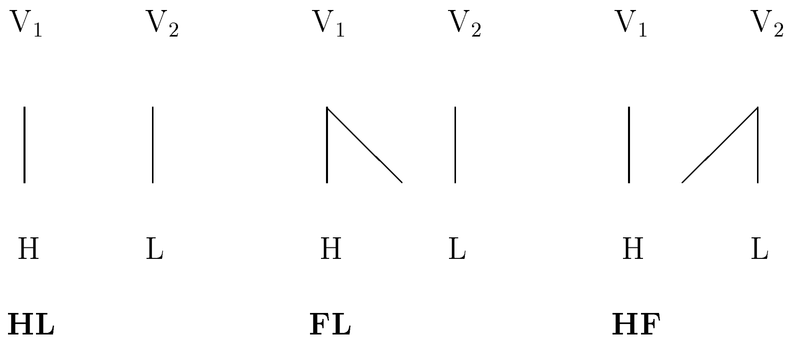
The tone pattern automata are depicted in (21).
Intersecting the stem representations with the automata we get the candidate sets in (23).

(23) Possible Stem Representations Including Tone Patterns

	L	**H**	**HL**	**LH**	**LHL**
1	**L**	**H**	**F**	**R**	**RF**
2	**LL**	**HH**	**HL HF FL**	**LH LR RH**	**LF LT RL RF TL**
3	**LLL**	**HHH**	**HLL HHL HFL FLL**	**LLH LRH LLH RHH**	**LHL LRL LFL LHF LTL LLT RHL RFL RHF TLL**

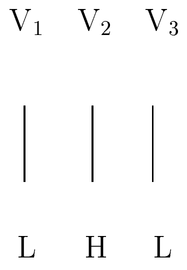
This corresponds for each pattern to the set of exhaustive possible associations without crossing, as is shown here for **HL** and **LHL**[9]:

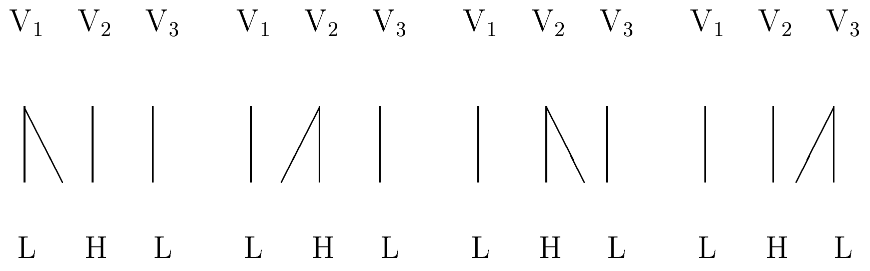
(24) Possible Associations for Two Syllables and **HL**



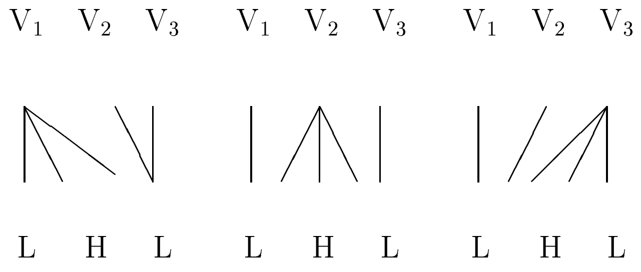
[9]For the **H** and the **N** pattern this is again trivially true, since there is only one exhaustive mapping. **LH** is again the mirror image of **HL**.

(25) Possible Associations for Three Syllables and **LHL**

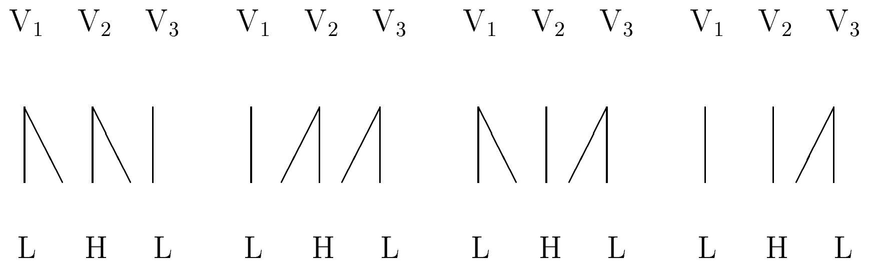




LHL **LRL** **LHF** **LFL**



LHL **LRL** **LHF** **LFL**



LHL **LRL** **LHF** **LFL**

4.3 The Well-Formedness Condition as a Finite-State Constraint

The characteristic effect of Goldsmiths mapping algorithm including the WFC is that contours and tone plateaus occur only finally in words, that is following alternating tones (**HLH**, ..., or **LHL**, possibly of length 0) and excluding each other. Further it ensures by its working that final contour tones always start with **H** (**F**) when a **L** tone precedes, and with **L** (**R, T**) after a **H** tone. All these effects can be achieved without association, simply by demanding it in form of a finite-state constraint:

(26) Well-Formedness Condition

$$\begin{array}{ll} \{StartL\} & L(HL)^*(L^* \mid H^* \mid F \mid HT) \\ \{StartH\} & H(LH)^*(L^* \mid H^* \mid R \mid T) \\ \{Contour\} & (R \mid F \mid T) \end{array}$$

$$(\{StartL \mid \{StartH\} \mid \{Contour\})$$

For illustration the table in (27) shows all tone patterns from 1-3 syllables that it allows.

(27) Well-Formed Tone Sequences

	starting with **L**	starting with **H**	starting with Contour
1	**L**	**H**	**F/R/T**
2	**LL LH LF**	**HH HL HR**	
3	**LLL LHL LHH LHT**	**HHH HLH HLL**	

Since we want to intersect (26) with the intersection of stems and tone patterns, the reader can find the correct tone pattern by "intersecting" tables (23) and (27), i.e choosing for each cell in (23) a candidate that can also be found somewhere in (27). E.g. the candidates for two-syllable **HL** stems are **HL, HF, FL**. The only pattern of these that can be found in 27 is **HL**. Of course the same results are obtained by the technical intersection of the automata in (21) and (26).

5 An Analysis Using Violable Constraints

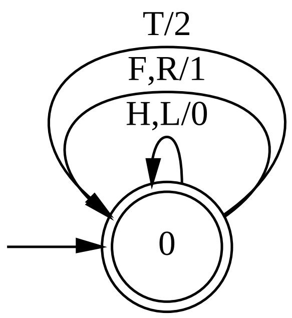
5.1 *Contour as a Violable Constraint

The revised well-formedness condition in (26) certainly allows a better understanding of the formal nature of Mende tone mapping, but it remains an ad hoc device without further independent support. In fact most of the data can be

(28) Relation of Syllable Numbers and Occurrence of Contours

Tone Pattern	N(syl) = N(tones)	N(syl) < N(tones)	N(syl) > N(tones)
H	*kɔ̈*		*pɛ̈lɛ̈* *häwämä*
L	kpa		*bɛlɛ* *kpakali*
HL	*ngïla*	*mbù*	*fëlama*
LH	*fandë*	*mbá*	*ndavülä*
LHL	*nïkïlï*	*mbâ* *nyahä*	

(29) *Contour



explained by very natural phonological constraints namely the ban on contour complexity in tones and the OCP, that applied to tones prohibits adjacent identical tones. This can be seen more clearly in the following table in (28) . Contour tones and plateaus arise only, when they can't be avoided, since the number of available syllables doesn't allow a 1:1 mapping of syllables and tones:

Note further that the more complex contour **T** (**LHL**) is avoided in favor of a two-tone contour. E.g **LHL** could be realized in nyahä i.a. also as **nyahâ* but only *nyahä* is correct. The notion of violable constraints familiar from Optimality Theory allows a natural implementation of the intuition that contour tones and plateaus are avoided when possible, but occur otherwise, and that there is a hierarchy of preference between simple two contour and three- contour tones . Such constraints can be realized by finite-state transducers mapping phoneme strings into $\{0, 1, 2, \ldots\}$, where numbers > 0 stand for constraint violations differing in weight and 0 for no violation. I'll illustrate this with a constraint marking contour tones, given in (29).

For sake of simplicity I again omit consonants. Like until now we can use the tone patterns and intersect them with the lexical representations. For a two-syllable word associated with **HL** we get the following results mapped by the transducer to constraint violations:

(30) *Contour-Violations for a Two-Syllable **HL**-Stem

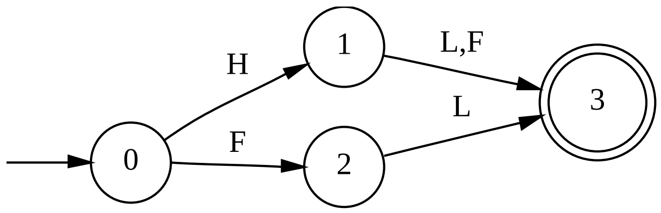
HL	HF	FL
00	01	10

Since the second and third alternative involve one constraint violation ("1") each and the first one none it is clear, that this is the optimal candidate, and hence following the principles of OT the correct one.

5.2 Local Constraint Evaluation

The method used here implicitly to find the optimal candidate is to generate all candidates and a complete evaluation of each, count the respective constraint violations for each, and then choose the candidate with the fewest violations as the correct one. This is a version of a global evaluation procedure. But in a finite-state framework there is a local alternative in terms of traversing automata by choosing at each state the optimal transition. A preliminary version of such an account will be shown to have desirable empirical consequences in this and the next section. A more formal version is developed in section 5.5. The candidate set for our example as a deterministic[10] FSA is :

(31) Candidate Set for a Two-Syllable **HL**-Stem



Now we can traverse the automaton and the evaluation transducer at once, choosing at each state the optimal transition, that is the one mapping onto the smallest number value.We will get the path $A - H - B - L \rightarrow E$, and hence the same result as above. So far global and local evaluation make no empirical difference, but there are cases where they do. Let's first see a hypothetical example:

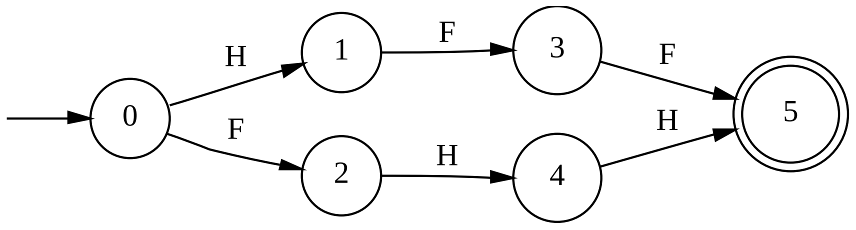
Globally the second candidate violating the contour constraint only once is clearly better than the first one violating it two times, but under the local evaluation procedure the first one is better, as can be seen with the corresponding automaton. In the first step the transition to B will be chosen, since it's locally the best, and then the only remaining path is $H - B - F - F \rightarrow F$.

[10]see the next section for a definition of deterministic.

(32) Hypothetical Tone Pattern Candidates

HFF	**FHH**
011	100

(33) Hypothetical Candidate Set as Automaton



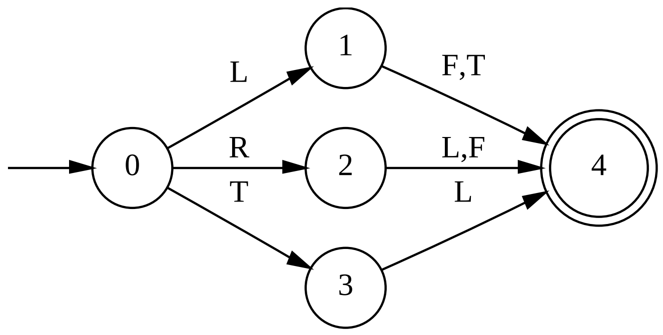
Examples of this type are a logical possibility, but empirically they don't occur in the Mende data. This is a first argument suggesting that the weaker local variant of constraint evaluation might suffice. But there's still stronger evidence. Look at the following candidate set for two syllables with the **LHL** pattern:

(34) Candidate Set for Two-Syllable **LHL**-Stems

Tone Pattern	**LF**	**LT**	**RL**	**RF**	**TL**
mapped to	01	02	10	11	20
Sum	1	2	1	2	2

Under global evaluation **LF** and ***RL** both are optimal, and there's no way to find the correct candidate. However **LF** is the only correct candidate under local constraint evaluation, as can bee seen from the corresponding automaton in (35).

(35) Candidate Set from (34) as Automaton

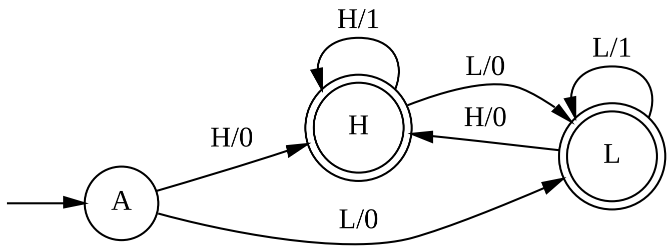


The first step of traversing the automaton in (14) can only be over **L** to state 1, since this means weight 0 in (29). Equally clear is the next step over **F** to state 4: 1 in (14) is better than 2. We obtain the correct **LF**.

5.3 Implementing the OCP

It's simple to implement a finite-state constraint that marks each tone that follows an identical tone, when we consider only non-contours:.

(36) OCP as Finite-State Transducer



But we certainly also want to put a ban on cases like **HF** (**H HL**), where the OCP violation occurs between the first tone and the first half of the contour[11], and anyway we have to integrate contours in the constraint if only to accept forms with contours. We can use an straightforward extension of the algorithm for integrating contours into tone patterns. The only difference is that we have to

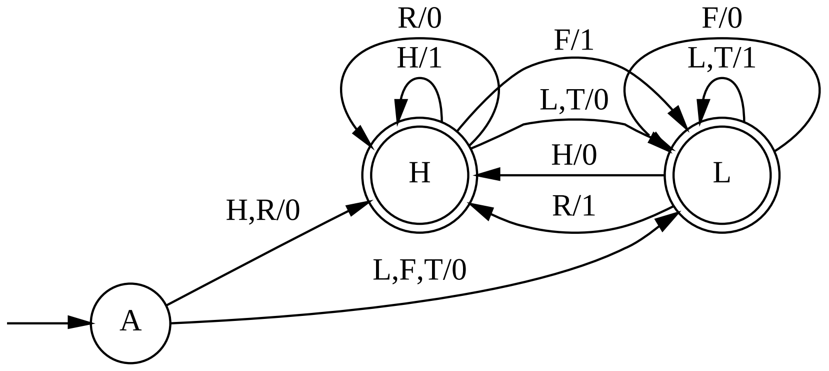
[11]For the Mende data it isn't technically necessary to mark OCP(Tone) violations inside of contours.

fix the second value of the resulting transitions, for which I will take the sum of the 'underlying' simple transition values. Let's see in tabular form which paths in (36) correspond to contours and which new transitions are introduced. (38) then gives the modified transducer:

(37) Additional Transition for Integrating Contours in (36)

Contour	Paths
F(HL)	$A - 0 \rightarrow H - 0 \rightarrow L$
	$L - 0 \rightarrow H - 0 \rightarrow L$
	$H - 1 \rightarrow H - 0 \rightarrow L$
R(LH)	$A - 0 \rightarrow L - 0 \rightarrow H$
	$H - 0 \rightarrow L - 0 \rightarrow H$
	$L - 1 \rightarrow L - 0 \rightarrow H$
R(LH)	$A - 0 \rightarrow L - 0 \rightarrow H - 0 \rightarrow L$
	$H - 0 \rightarrow L - 0 \rightarrow H - 0 \rightarrow L$
	$L - 1 \rightarrow L - 0 \rightarrow H - 0 \rightarrow L$

(38) OCP as Finite-State Transducer (with Contours)



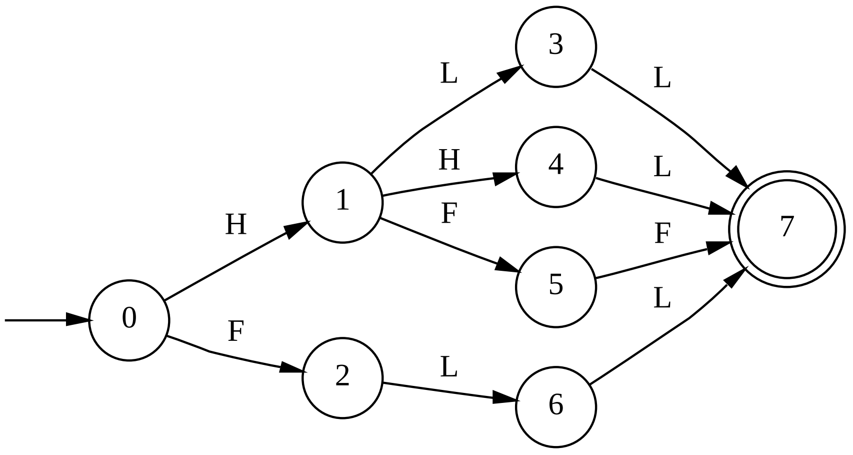
As you can see easily **R** (**LH**) and **T** (**LHL**) after **L** and **F** (**HL**) after **H** are correctly mapped to 1. This shows that the chosen strategy of introducing contours into constraints by (meta)operations on automata. not only works in mapping tones to segments, but also in contextually working constraints. Let's now show that OCP(Tone) has to be interpreted locally to get the right results. The candidates for a three-syllable word with **HL** patterns are:

Independently of how we weight against each other the two constraints, **HLL** and **HHL** are the best candidates and equally good or bad under global constraint evaluation. Traversing the corresponding transducer and OCP(Tone)/*Contour on the other hand correctly selects **HLL**:

(39) Candidate Set for a Three-Syllable **HL**-Stem

Pattern	**HLL**	**HFL**	**HHL**	**FLL**
OCP-Violations	001	011	010	011
*Contour-Violations	000	010	000	100

(40) Candidate Set from (39) as Automaton



The first step will be to state 1 on **H** since **F** maps to 1 on *Contour, while **H** maps in both constraints to 0. The second step goes on **L** to [2] avoiding the 1-mapping on **L**, and for the last step only one possibility remains (**L**).

5.4 Deriving the Whole Range of Data

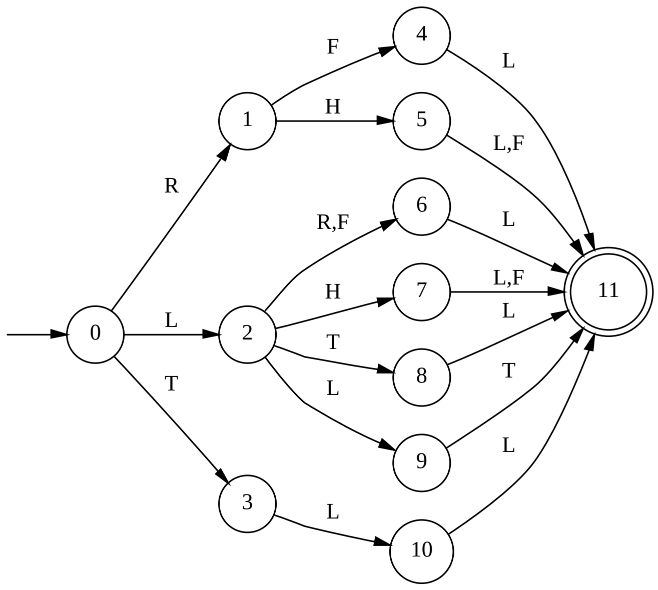
As a starting point we take again the table in (23). The task now is to find the optimal candidate for each cell. Since for cells with only one candidates this candidate will always be optimal we don't have to say anything about the **L** and **H** patterns or the stems with one syllable. **LH** is again the mirror image of **HL** so it suffices to consider the following data:

(41) Relevant Patters from (23)

	HL	**LHL**
2	**HL HF FL**	**LF LT RL RF TL**
3	**HLL HFL HHL FLL**	**LHL LRL LFL LHF LTL LLT RHL RFL RHF TLL**

We have already shown how to derive the correct results for the **HL** data and for two syllables with **LHL**. So the only case that remains is **LHL** with three syllables. The candidate automaton is:

(42) Candidate Automaton for Three-Syllable **LHL**-Stem



Transversal starts with $0 - L \rightarrow 2$, since it's the only non-contour, and OCP(Tone) for all transitions gives 0. $2 - H \rightarrow 7$ is optimal next: all other alternatives are contours or/and violate OCP(Tone) while **H** implies no violation of either constraint. In state 9 **F** would violate *Contour and OCP, **L** none, thus **L** is chosen.

5.5 Algorithms

The procedure that was used in the previous sections to describe local constraint evaluation is problematic in that it doesn't guarantee termination[12] and presupposes that constraint evaluation always results in only one optimal candidate, since no provision was made for multiple optimal transitions from single states. The following algorithm retains the basic intuition of locality, but avoids these shortcomings.

[12]We might find us in the candidate automaton and in the transducer in non-final states with only one outgoing transition leading to the respective states themselves. This transition would be optimal and would be chosen over and over again.

Algorithm 2 Local_Constraint_Evaluation(A, T) = **LCE**(A, T)

```
1: $T_1 \Leftarrow$ Left_Restriction$(A, T)$
2: $T_2 \Leftarrow$ Optimize$(T_1)$
3: LCE$(A, T) \Leftarrow$ Left_Language$(T_2)$
```

Left_Restriction combines a candidate automaton and an evaluation transducer and generates in effect the set of strings described by the automaton annotated with the violation marks from the transducer:

Algorithm 3 Left_Restriction$(A, T) = A \cap_{Left} T$

```
1: make $(I_A, I_T)$ initial in $A \cap_{Left} T$
2: make $(F_A, F_T)$ final in $A \cap_{Left} T$
3: for each arc from $z$ to $t$ in $T$ labeled $M_2/P$ do
4:    for each arc from $x$ to $y$ in $A$ labeled $M_1$ do
5:       if $M_1 = M_2$ then
6:          add to $A \cap_{Left} T$ an arc from $(x, z)$ to $(y, t)$ labeled $M_1/P$.
7:       end if
8:    end for
9: end for
```

Optimize deletes all transitions from the transducer which aren't optimal in a particular state:

Algorithm 4 Optimize(T)

```
1: Input: $T$, a finite-state transducer
2: Output: Optimize$(T)$, a finite-state transducer
3:
4: make $I_T$ initial in Optimize$(T)$
5: make $F_T$ final in Optimize$(T)$
6: for each state in $T$ $S_i = S_1$ to $S_n$ do
7:    for each transition from $S_i$ $T_j = T_1$ to $T_m$ do
8:       if $V_j = Min_{1-m} V_j$, where $T_j = X_j/V_j$ then
9:          add $T$ to Optimize$(T)$
10:      end if
11:   end for
12: end for
```

Finally, **Left_Language** generates an automaton by removing the respective second symbol from each symbol pair[13]

A further advantage of the formalization in algorithm 4 is that it can be used in a standard way (cf. Eisner, 1997a) to implement OT-like constraint-

[13]For a formal definition see Frank and Satta (1998).

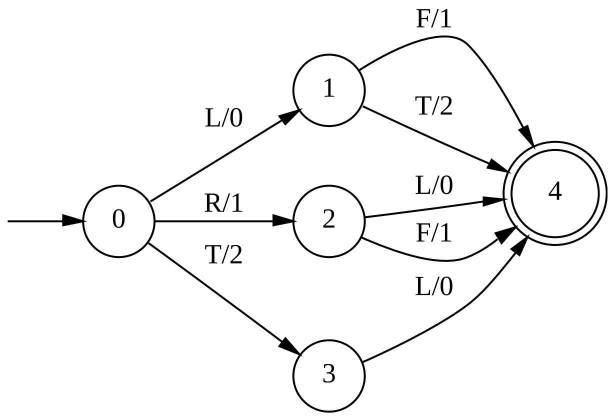
hierarchies. Since the algorithm for a given candidate automaton *Cand* and a constraint *Cons* generates a new automaton **LCE**(*Cons*, *Cand*) containing only the optimal candidates w.r.t. *Cons*. The automaton **Opt**($R, Cand$) containing the optimal candidates w.r.t. a candidate set *Cand* and a Constraint ranking $R = C_1 \ldots C_n$ can then be determined recursively as follows:

Algorithm 5 Optimal($R, Cand$)

```
1: Opt($R, Cand$) ⇐ $Cand$, if $R$ is empty
2: Opt($R, Cand$) ⇐ Opt($R', Cand'$), otherwise,
   where $R' = C_2 \ldots C_n$ and $Cand' =$ LCE($C_1, Cand$)
```

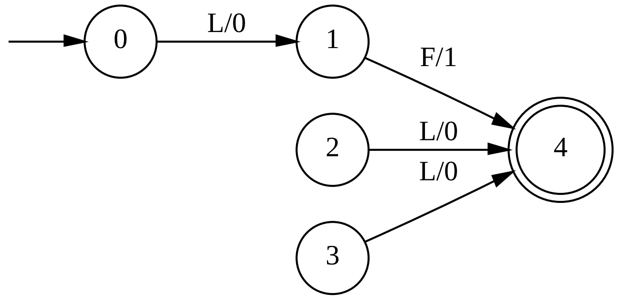
Let's see how the new version can derive the correct result for a two syllable stem with **LHL**, i.e. we compute **LCE**(A, T), where A is the candidate automaton in (35) and T is the *Contour-Transducer in (29). **Left_Restriction**(A, T) gives:

(43) Two-Syllable **LHL** Automaton $\cap_{Left}$ *Contour-transducer



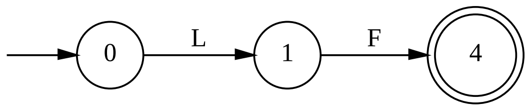
Removing all suboptimal transitions results in

(44) **Optimize**(43)



After deleting the dead states 2 and 3 and the transition values we arrive finally at the expected

(45) **Left_Language**(44)



6 Degrees of Explanatory Force

6.1 The Auto-Segmental and the Declarative Account

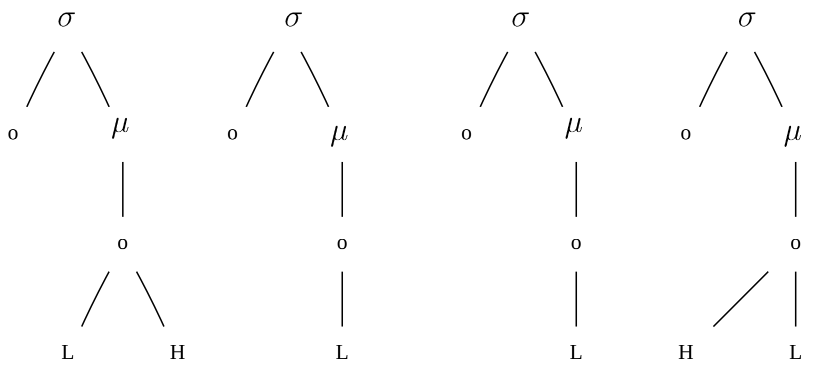
In a sense it's trivial to account for the Mende data using a finite-state device. Since there is only a final number of noun stems, and for each final string set there's a finite-state automaton accepting it, there certainly is one accepting all and only the Mende noun stems, including the correct tone specifications. But Goldsmiths account goes one step further. It modularizes tone patterns and their concrete realizations in stems with different syllable number. Thus it predicts that given the 5 basic tone patterns in stems there will occur only the observable 25 tone realizations. These advantages carry over to the two models introduced here. Moreover, and this is seen most clearly in the first model it becomes clear that the tone mapping rules given by Goldsmith are dispensable, when we conceive tone pattern and well-formedness condition as independent partial descriptions of the same object. On the other hand the well-formedness condition in (26) is completely ad hoc, while Goldsmith uses a mechanism proven useful in other domains like Semitic Root- and Pattern-Morphology (cf. McCarthy, 1979) and auto-segmental accounts of reduplication (cf. Marantz, 1982). But even if applicable to diverse phenomena the mapping algorithm remains stipulation, and the second model proposed here goes beside this by reducing it's effects to phonological constraints and the locality of constraint evaluation. Thus, if evaluation locality can be maintained in somewhat like the form described all that remains as really specific to the Mende data are the specific tone patterns and this seems at the current point to be the absolute minimum to account for the data in an explanative way.

6.2 Standard OT-Accounts

Standard-OT accounts of tone have abandoned procedural devices like Goldsmiths tone mapping algorithm in (4) and rely on constraints of the Generalized-Alignment-Type (McCarthy and Prince, 1993). In this section I will briefly discuss Heiberg (1999)s analysis of the Mende data and compare it to the account working with local constraint evaluation. Heiberg accounts for the fact that tone realizations without contours are generally prefered in much the same way as we did, namely by a specific high-ranked constraint, which she calls "**GROUND-NEGATIVE**(H,L)".

Two further constraints (**AlignPRWD**(H, Right), **AlignPRWD**(L, Right)) require that the (time unit at the) right edge of the prosodic word be associated with **H** and **L** respectively. As long as corresponding constraints for the left edge of the word are ranked lower this has the effect of attracting occuring contours (double association) to the right since this is the only possibility of getting **H** *and* **L** in association with the rightmost time unit. as is illustrated by the representations in (46).

(46) Representations for *nyáha* and *nyahà* (Heiberg, 1999, pg. 93)

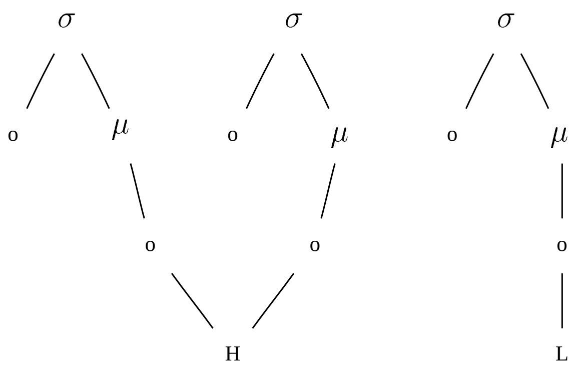


Finally, tone plateaus tend to the right, because of **ALIGNFEATURE**(H,left) and **ALIGNFEATURE**(H,right), which punish all tones that intervene between the leftmost association of the respective tone and the left word edge. As can be seen in (47) plateaus at the left word edge maximize the relevant distances while right edge plateaus minimize them.

Heiberg's account is problematic in several respects. First, it relies extensively on Generalized-Alignment constraints, which are questionable in introducing extensive constraint parameterization and are computationally the most complex constraint type in the OT-literature (cf. Eisner, 1997b; Heiberg, 1999, pg. 181), The account in this paper on the other hand uses evaluation procedures for constraints that are actually less complex than the ones usually proposed.

Second, Heiberg's proposal predicts a large variety of possible tone mappings (in different languages), which are un-attested. For example a ranking of the form **ALIGNFEATURE**(H,left) ≪ **ALIGNFEATURE**(L,right) ≪ **ALIGNFEATURE**(H,right) ≪ **ALIGNFEATURE**(L,left) should result in a language where **H** plateaus go to the right word edge, and **L** plateaus to the left. Similarly we should find languages where contours tend to the right edge, while plateaus appear at the left, which would be accomplished by **ALIGNPRWD**(H, right), **ALIGNPRWD**(L, right) ≪ **ALIGNFEATURE**(H,left) ≪ **ALIGNFEATURE**(L,right). A ranking where **ALIGNPRWD**(H, right), **ALIGNPRWD**(L, right), **ALIGNPRWD**(H, left), **ALIGNPRWD**(L, left) are the

(47) Representations for *felamä* and *fëlama* (Heiberg, 1999, pg. 95)


σ
σ
σ
o
μ
o
μ
o
μ
o
o
o
H
L