On Finite-State Tonology with Autosegmental Representations

Anssi Yli-Jyrä
University of Helsinki, Department of Modern Languages
PO Box 24, 00014 University of Helsinki
anssi.yli-jyra@helsinki.fi

Abstract

Building finite-state transducers from written autosegmental grammars of tonal languages involves compiling the rules into a notation provided by the finite-state tools. This work tests a simple, human readable approach to compile and debug autosegmental rules using a simple string encoding for autosegmental representations. The proposal is based on brackets that mark the edges of the tone autosegments. The bracket encoding of the autosegments is compact and directly human readable. The paper also presents a usual finite-state transducer for transforming a concatenated string of lexemes where each lexeme (such as "babaa|HH") consists of a segmental substring and a tonal substring into a chronological master string ("b[a]b[aa]") where the tone autosegments are associated with their segmental spans.

1 Introduction

In Bantu linguistics, *Autosegmental (AS) Phonology* (Goldsmith, 1976) is a standard theory in phonological description of tone. The widely available finite-state compilers are, however, not directly applicable in this context because autosegmental phonology uses a two-tier representation for the phonological content. The aim of this paper is to address this obvious shortcoming in finite-state technology. I will, therefore, pursue a practical approach that facilitates conversion of an existing multi-tiered lexicon and an autosegmental rule system into a lexical transducer.

In the past, various finite-state approaches to autosegmental phonology have been proposed. Kay's (1987) early proposal about processing multilinear structures with an extended finite-state transducer model has inspired further research on multi-tape automata (Wiebe, 1992) and linear codes (Kornai, 1995) that encode events when an autosegmental representation is scanned from left to right. Kornai (1995) has qualified the proposed codes with a set of desiderata. All these desiderata cannot be, however, fully satisfied by any of the linear codes (Wiebe, 1992). An alternative to these multi-tape approaches is proposed by Bird and Ellison (1994) who posit that all tiers are partical descriptions of the common synchronized structure and they can, therefore, be combined via intersection. This constraint-based approach is very natural and it has nice formal properties such as declarativeness and connection to logic. However, the resulting one-level phonology (Bird and Ellison, 1994) is also somewhat incompatible with the autosegmental theory. For example, it does not posit floating tones that are a crucial formal device in many existing accounts of Bantu tone.

The key idea in this paper is to represent the tone units, i.e., autosegments, as time spans that have a start and an end marked with brackets in the timing tier. The key idea is complemented with a finite-state technique for producing tone associations from the lexical forms and a new, tailored finite-state formalism for autosegmental alternation rules. The implementation of each alternation rule is based on declarative, constraint-based techniques.

The resulting formalism can be seen as a reimplementation of the classical two-level formalism (Koskenniemi, 1983). The new formalism allows the user to specify even complex parallel changes to the autosegmental representation in a compact way. The reimplementation is based on generalizations that support parallel compilation of rules (Yli-Jyrä and Koskenniemi, 2006; Yli-Jyrä, 2008a) and the new, lenient semantics of obligatory rules (Yli-Jyrä, 2008b). The bracketing that I use to represent tone is reminiscent of foot

Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing, pages 90–98,
St Andrews–Sctotland, July 15–17, 2013.

bracketing (Karttunen, 2006), syllable bracketing (Idsardi, 2009) and the bracketing of tone domains (Leben, 2006), all representing Optimality Theoretical (Prince and Smolensky, 2004) approaches to phonology.

2 Theories of Tone

Tone in phonology relates to such a pitch contrast that distinguishes the meanings of two word forms. There are level tones such as: High (H), Low (L) and Mid (M), and contour tones such as: Rising (LH), Falling (HL), and Rising-falling (LHL), Mid-hig (MH), Mid-low-high (MLH). A segment (a mora or syllable) having the tone feature is called a *tone bearing unit (TBU)*.[1]

Williams (1976) and Leben (1973) distinguish and compare three different theories that claim to describe the nature of tone:

- the *segmental theory*,
- the *syllabic theory*, and
- the *morphemic theory*.

This taxonomy gives use a good starting point for explaining how suprasegmental and autosegmental phonology differs from the segmental phonology. It is important to note that the newer theories are improved generalizations inspired by the former theories rather than completely opposed to them.

2.1 Segmental and Syllabic Theories

Features and Natural Classes It would be the simplest theoretical solution to process sounds as segments having tonal features. Then the treatment of tone would not differ from any other features such as nasality or openness. Sounds and their classes would be viewed as a Boolean algebra and the feature geometry would define what sound classes or their changes are natural and what are not. In finite-state phonology, the natural classes, such as "V [+ High]" can generally be implemented as character lists, but Bird (1994; 1995) and Heinz and Koirala (2010) go much further in capturing the notions of feature geometry by computational means.

The segment-based representation of tone gives a clumsy treatment to *floating tones* that are not carried by any segmental units. The same weakness is true for syllabic theory: there are floating tones that are not linked to any syllable (Leben, 1973).

Subsegmental Diacritics as Segments A slightly more modular solution would treat tone as a diacritic character that affects the adjacent character. This is how UNICODE or the IPA standard handle tone at the character level. Likewise, some computational morphologies use segments such as H or L to represent the tone in the morphophonological representation. Muhirwe (2010) uses this in his finite-state description of Kinyarwuanda:

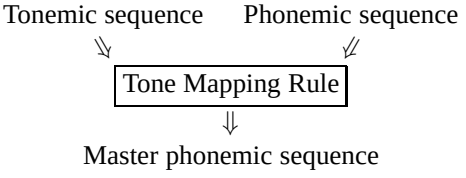
(1) *baHzaHaHriHriimba*<*ba-zaHa-riHriimba*
REL.FUTURE
'they might sing'

2.2 Morphemic Theories

In a morphemic theory of tone, the tone of underlying morphemes is indicated separated from rather than pre-assigned to the phonemic sequence. This separation is indicated in the lexicons of some descriptive grammars, such as (Halme, 2004):

(2) **elifikameno** *n LHHHL 5* independence.
elifiyepo *n LHLH 5* contest.
elifo *n LH 5* likeness.

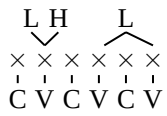
Suprasegmental Phonology According to Williams (1976) and Leben (1973), tones are suprasegmental. They are not marked underlyingly to the segments, but they constitute units in their own right. To merge the phonemic and tonemic sequences of morpheme, Williams proposes a Tone Mapping Rule (3).

(3)


A shortcoming of the original formulation of the Time Mapping Rule is that its output representation is purely linear, which complicates the description of further phonological processes. Therefore, Goldsmith (1973) argues that Leben's theory does not adequately describe some effects of floating tone.

[1] For simplicity, this paper assumes that each vowel is the tone bearing unit.

Multi-Tiered Representation The Autosegmental Phonology (AST) (Goldsmith, 1976) claims that what we see in the segmental representations of the language is an image of a richer multi-tiered representation that involves simultaneously pronounced sequences (Goldsmith, 1973). One tier represents the tone patterns via H/L tonemes, called *autosegments*, while another tier represents the usual C/V segments in the pronunciation. These two underlying tiers are integrated through the timing tier that consists of a sequence of ×'s:

(4)


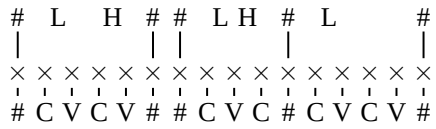
Tones are not necessarily in a one-to-one relationship with the segments. Instead, the nature of tone can be suprasegmental or subsegmental and therefore it cannot be incorporated to or ordered among the segments.

A two-tiered morphemic representation avoids the loss of tonemic structure. The tone associations (links) simplify the rules considerably and let the AST posit a simple, elegant theory of operations: add/delete a tone/link. This elegancy is perhaps the most attractive aspect of AST.

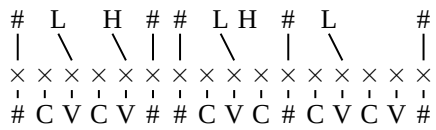
2.3 Autosegmental Derivation

Goldsmith (1976) defines the autosegmental derivation as follows:

Step 1: Initialization The phonological and tonological sequences are set out as parallel strings and the tone boundaries[2] are associated with each other.

(5)


Step 2: Association Rule (AR) The ASSOCIATION RULE associates the first toneme with the first tone bearing unit and proceeds rightward as long as the segments match one another.

(6)


[2]In this paper, we will naively assume that all morpheme boundaries are also tone boundaries.

Step 3: Well-Formedness Condition The well-formedness condition makes sure (i) that every toneme has a corresponding segment, (ii) that every phoneme segment is linked to a tone, and (iii) that the links do not cross one another. The changes required to enforce the well-formedness condition are unambiguous after the AR. At this point, only the rightmost member of a tier can be associated with more than one member of another tier of the word.[3]

(7)

Step 4: Mutations The phonological derivation with autosegmental rules now starts.

3 The Bracket Encoding of Associations

The encoding of autosegmental representations proposed in this paper is based on brackets that indicate the span of each tone autosegment in the segmental string. Instead of marking the tones separately, we only mark their links – in fact the first and the last link only – to the segmental tier. This means that the general bracketing approach familiar from e.g. Leben (2006) and Idsardi (2009) is now developed further and applied to non-OT representations of tone.

3.1 Common Timing Elements of All Tiers

The timing tier synchronizes the segmental and tonal tiers using morphological boundary symbols.

- the prefix/suffix boundaries (`-`/`+`)
- named affix boundaries (such as `AUG-`).
- the clitic boundary (`=`)
- the infix boundaries (`<IFX` and `IFX>`)
- the reduplication boundary (`~`)
- the word boundary (#)
- morphological categories (such as `N5`).

These boundaries are shared both by the segmental and tonal tiers. Thus there is no need to *add* separate inter-tier associations for the morpheme boundaries.

[3]On different assumptions on tone association rule and tonal boundary markers, see Leben (1978).

3.2 Segmental Tier

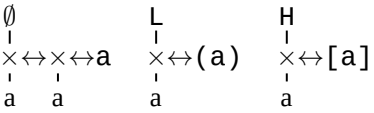
Segmental tier consists of segmental phonemes – vowels (V) and consonants (C). The segmental tier may also contain the syllable boundary (.).[4]

The foot structure can be marked with additional types of syllable boundaries .(. , .). and .)(. that are considered as multi-character symbols, in contrast to Karttunen (2006).

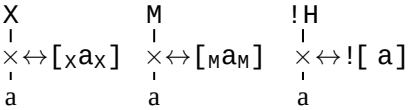
3.3 Tone Associations

There is usually only a small number of underlying tone level distinctions such as: Toneless (∅), Low (L), and High (H). As to the surface tones, the international phonetic alphabet lists, for example, five level tones and suggests many more contour tones. However, it is now not necessary to go through all the theoretically possible surface tones that might occur in the languages of the world.

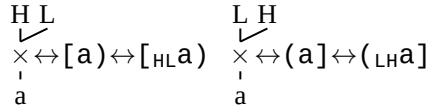
The key proposal of this paper is to indicate the association of tone autosegments by showing the span of the tone via brackets.



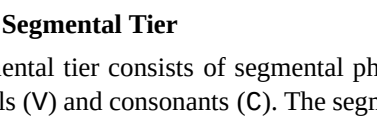
In the lexical representation, an unspecified tone can be marked with X. When the lexical tones undergo changes, the tonal tier can also contain additional tones such as Middle (M) and Downstepped High (!H):



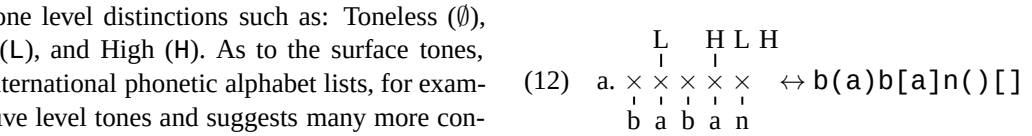
The basic contour tones, such as H͡L and L͡H, are notated by mixing the brackets of two different tones. To facilitate notating more complex contour tones, we can introduce labeled brackets for simpler contour tones:



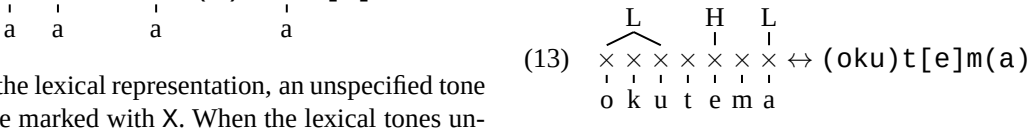
Contour tones with three underlying tones are represented using simpler contour tone brackets as needed:



Floating tones do not contain any vowels in their spans. If a floating tone emerges due to the well-formedness condition after the association rule, it will be placed immediately before the next tone (i.e., morpheme) boundary. In all other cases, the place of the floating tone in the segmental tier is specified by its derivation history.[5]



We posit a convention according to which the brackets of a linked tone will be inserted immediately around the linked segments, without any intervening boundary symbols, syllable boundaries and stress markers. If more than one segment are linked to one tone, the brackets span all the linked segments:



4 Derivation of the Input for the Rules

In oder to use the bracketed representation, we need to implement the first three steps of the autosegmental derivation.

4.1 Specifying the Task

The first derivation step takes the lexical form as its input and produces a string where the tones are associated with the segments. The underlying lexical form contains at least the tonal string, the phonemic string and morphological boundaries. In practice, it is convenient to have also some information on the morphological categories and the glosses in the morpheme lexicon. For example, CL5 and N5 in the glossed underlying string (14) are morphological category labels and 'likeness' is the semantic gloss of the stem.

[4] Syllable stress markers such as ′ could be added near the syllable boundaries, but we try to avoid going into the stress structure in too much detail in this work.

[5] We will need to experiment more before we know if there is a need to normalize the floating tones. Meanwhile, it would seem natural to migrate the brackets of floating tones as little as possible.

Table 1: The *foma* Syntax and Basic Definitions

`0`	empty string
`"|"`	protected special symbol
`{abc}`	a string of letters
`?`	all (identity pairs of) symbols
`A B`	concatenation
`A|B`	union
`[A]`	grouping
`(A)`	optionality
`A*, A+`	Kleene star, Kleene plus
`A&B`	intersection of 0-padded relations
`A-B`	difference of 0-padded relations
`A:B`	left-aligned cross product
`A.x.B`	left-aligned cross product
`A.o.B`	composition
`A.i`	inverse relation
`A.1, A.2`	input projection, output projection
`A/B`	free insertion of symbol pairs `B`
`def A B;`	definition of a constant expression
`def A(X) X X;`	definition of a regex macro

Phonemic (Segmental) Tier

```
def V       a | e | i | o | u ;
def C       b | d | ... | y ;
def S      "." | ".(." | ".)." | ".)(." ;
def M      "-" | "AUG-" | "+" | "#" | "::";
```

Tonemic (Autosegmental) Tier

```
def L       "(" | "[" | "![" | "[LH" | "[HL" ;
def R       ")" | "]" ;
def T       L | R ;
def X       L:L | R:R | T:0 | 0:T;
```

(14) **elifo** *n LH 5* likeness. (Halme, 2004)

 # e- lifo|LH *#*

In a finite-state implementation, the underlying form (14) can be written as string (15) where `::`, `CL5`, `N5` and `'likeness'` are multi-character symbols. In this string, the double colon `::` indicates the glossing relation.

(15) `#e::CL5-lifo|LH::N5'likeness'#`

The task of the first three steps of the autosegmental derivation is to implement the mapping (16a) or just to produce the output of the mapping (16b). The output will then be the autosegmental representation fed to the autosegmental alternation rules.

(16) a.
```
#e::CL5-l i f o |LH::N5'likeness'#
#e::CL5-l(i)f[o]   ::N5'likeness'#
```

b.
```
#e::CL5-l i f o |LH::N5'likeness'#
#e     -l(i)f[o]                 #
```

4.2 The Implementation Formalism

We will use the regular expressions of the freely available *foma* tool (Hulden, 2009) when implementing all the rules. This flavour of regular expressions has been originally developed at Xerox. The relevant syntax of the formalism is summarized in Table 1.

Table 2: The Definition of the Association Rule

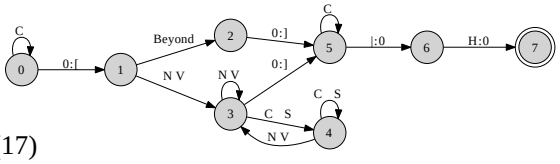
```
def Tones         "L" | "H" ;
def Mapper(TBUs) [ "L"  .x. "(" TBUs ")" ] |
                 [ "0"  .x.     TBUs     ] |
                 [ "H"  .x. "[" TBUs "]" ] ;
def Asso(Pat,TBUs) [ [?-T] | 0:T ]* .o.
        [ Pat .o. [0:C | Mapper(TBUs)]* ].2 ;
def Single   V | Beyond ;
def TheRest  V [[C|S]* V]* | Beyond ;
def Map(Pat) Asso([Pat .o. ?* ?:0 ].2, Single  )
             Asso([Pat .o. ?:0* ? ].2, TheRest )
              "|":0  Pat:0 ;
def Maps     Map({LH}) | Map({LLH}) | Map({LHL}) |
             Map({0H}) | Map({00H}) | Map({0H0}) ;
def AR   [ [ ? - Beyond] | 0:Beyond* "|" ]* .o.
         [ ?* M |        Maps M ]* .o.
         [ [ ? - Tones - Beyond ] | Beyond:0 ]*;
```

Using the regexp syntax, we first define frequently used constant expressions for the rules. These expressions define symbol sets used in Phonemic and Tonemic tiers in Table 1.

4.3 Implementing the Association Rule

A crucial assumption in this paper is that the tone patterns can be stored into a finite-state memory. For each tonemic sequence `Pat`, I construct a transducer `Map(Pat)` that assings the sequence of tones to all the tone bearing segments of a morpheme and then removes the lexical tone pattern from the end of the morpheme. If there are fewer tones than TBUs, the last one will be spread over the rest. More concretely, the expression `Map({H})` is compiled into the transducer (17). A finite union of such transducers is stored under the name `Maps`.

(17)



The expression `AR` is a transducer that applies the `Maps` transducer to every morpheme in the input. In its definition, a hidden symbol, `Beyond` is used as a temporary TBU for tones that are left over. This implementation of the AR synthesizes the first three steps of the autosegmental derivation into one transducer.

After the appropriate language-specific changes to the tier definitions and the `Maps` transducer have been made, we can use the *foma* command line in-

terface to see how some example strings are processed by the AR rule:

```
(18)  : regex {#olu|LL-vala|HH#}.o.AR;
      : print lower-words
      #(o)l(u)-v[a]l[a]#
```

5 The Tone Alternations

The autosegmental rule formalism is based on rewriting rules. However, the input and the output are not just strings but linked pairs of strings. Most rules are notated using a shorthand convention according to which the input and output representations of the rule are combined into one representation where the tone delinking, deletions, insertions, replacements, and linking are indicated.

In this section, every rule will be written in two ways: with the original autosegmental notation and with regular expressions.

5.1 The New Rule Formalism

The rules will be expressed as regular expressions that describe the changes and the context conditions under which the changes can take place. The currently available definitions of the formalism are listed in Table 3.

Table 3: Formalism for bracketed rules

notation = foma expr.	**meaning**
L,R,T	sets of tone brackets
S,M	sets of syll., morph. boundary
V,C	sets of vowels and consonants
A=?-"<>"	any symbol
α_U=α	protected constant tone bracket α
($_U$	example: constant tone bracket (
α_I=X&[α|α:T|α:0]	tone bracket α in input
α_O=X&[α|T:α|0:α]	tone bracket α in output
α_X=α_I|α_O	mutable tone bracket α
α_C=[X& T:α]-T	changed tone bracket α
Q=[C|S]*	consonants and syll. boundaries
N=[C|S|M]*	anything constant but T and V
P=[C|S|M|V]*	anything constant but T
Z=[A|T$_X$]*	anything, including the mutable
α_D="<>"* α:0	delete bracket α
α_A="<>"* 0:α	add bracket α
α_F="<>"* α_O	enforce output bracket α
α_M="<>"* α_C	mutate bracket α
α:0	test for deleted α
$\overline{\alpha}$=Z-α	negates the expression α
α*	iterate α zero or more times
α+	iterate α one or more times
(α)	make α optional
[α]	grouping
α|β	α and β are alternatives

5.2 Example Rules

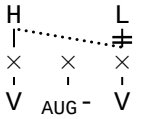
Anyanwu (2008) classifies some universally accepted tonal rules. Given such a classification, we consider some example rules from Kwanyama (Halme, 2004) and Ikoma (Aunio, 2010) and extend the classification where needed. By compiling different kinds of rules into the new notation, we get an estimate on out how the bracket-based formalism applies to the practical needs in general.

5.2.1 Spreading

The TONE SPREADING rules affect the following tone (e.g. LHH → LLH). If the spreading effect is partial, this results into a contour tone as in (e.g. LH → LL̂H) (Anyanwu, 2008).

AUGMENT HIGH SPREAD (AHS) of Kwanyama (Halme, 2004) causes the underlying High of the augment prefix to spread onto the following Low-toned mora (dotted line) whose tone is delinked (cut line).

(19) a.



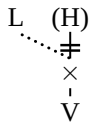
b. Z]$_D$ N $_{AUG}$- N ($_D$ V]$_A$ ($_A$)$_U$ Z

5.2.2 Assimilation

The TONE ASSIMILATION rules can be either regressive (HL → ML) or progressive (LH → LM) (Anyanwu, 2008).

HIGH LOWERING (HL) of Ikoma (Halme, 2004) causes a floating Low tone to link (dotted line) to the following mora whose High tone is delinked (cut line) and deleted (parentheses):

(20) a.



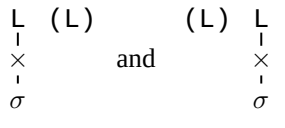
b. Z ($_D$)$_D$ N ($_M$ V)$_M$ Z |
Z ($_D$)$_D$ N ($_M$ V)$_A$ N [$_A$ V Z

5.2.3 Simplifications

The TONE ABSORPTION rules (e.g. L̂HH → LH) simplify two adjacent identical tones (Anyanwu, 2008). This rule is motivated by the OBLIGATORY CONTOUR PRINCIPLE (OCP) (Leben, 1973) that bans two consecutive features in the underlying representation.

FLOATING LOW DELETION (FLD) of Kwanyama (Halme, 2004) occurs when a floating

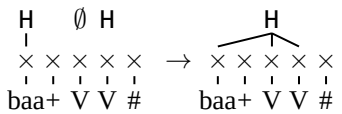
Low occurs next to a linked Low:

(21) a. 

b.
```
Z V )U N (D )D Z |
Z (D )D N (U V Z
```

In contrast to the TONE ABSORPTION rules, the CONTOUR LEVELING rules make two adjacent tones similar by simplifying a contour tone. (e.g. HL͡H → HH) (Anyanwu, 2008).

PLATEAUING in Ikoma (Aunio, 2010) is a variant of this kind of simplification:

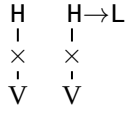
(22) a. 

b.
```
Z ]D "-" N V N [D V ]U N "#" Z
```

5.2.4 Dissimilation

DISSIMILATION (HH → HL) and TONAL POLARIZATION (HX → HL) are rules that are motivated by the OCP because they differentiate adjacent tones.

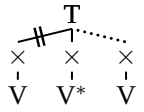
MEEUSSEN'S RULE in Ikoma (Aunio, 2010) lowers the last H in a sequence of two HH's:

(23) a. 

b.
```
Z ]U N (M V P )M Z
```

5.2.5 Tone Shift

TONE SHIFT (TS) in Kwanyama (Halme, 2004) moves all tones one mora (TBU) to the right (HLHL → ∅HLH). The correct interpretation of the rule assumes that there is no floating tones. [6].

(24) a. 

b.
```
Z (I V (RX) N (F V Z |
Z [I V (RX) N [F V Z |
Z LI V (RX) N LD V Z |
Z LO (P|T:0)* RD Z |
Z RI N (LX) V RD P (LX) V Z |
Z )I N (LX) V )F (LX) P Z |
Z ]I N (LX) V ]F (LX) P Z
```

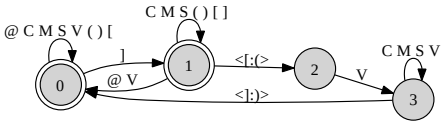
[6]The last shifted tone could land to a final position, but the current formulation does not support this. The floating tones can be shifted and produced using temporary tone bearers during the rule application as we did in the TONE ASSOCIATION RULE

5.3 The Rule Compiler

My *foma* code for the rule compiler is given in Table 4. This compiler consists of two parts:

- The macro `CR(Rule)` produces a transducer that contains the final rule transducer as its subset.
- The second macro, `COERCE(Rule)`, produces the final rule transducer by restricting `CR(Rule)` in such a way, that it performs, in a sense, as many individual changes as it can.

The first macro, `CR(Rule)`, works as follows. The basic compilation of the regular expression corresponding to an autosegmental rule notation (such as 23b) yields a 0-padded transducer, `Rule`, where an optional, freely iterated marker `"<>"` has been added at the positions where the input string is expected to change.[7] Another transducer, `W`, contains all possible string pairs `Z` into which we have added the marker for an arbitrary change concerning a tone bracket. The purpose of this transducer is to tell that the marked change requires a permission from rule to be acceptable. When these two 0-padded transducers are differentiated, the resulting transducer contains all those string pairs that have an unwanted change. The complement of this 0-padded transducer with respect to `Z` is then exactly the transducer whose string pairs contains only such changes that are specified by the rule. For MEEUSSEN'S RULE, this transducer is (25).

(25) 

The second macro, `COERCE(Rule)`, refines the transducer computed by the first macro. This macro marks, in each 0-padded string pair, all those positions where a bracket changes. The markup is done, again, using the same marker symbol, `"<>"`, as before. The input projection of this transducer gives a regular language without the zeros that were used in the transducer. An auxiliary macro, `TooFew(X)`, is now used to find out in this projection such marked input strings that are like some other string in the projection but contain markers only in a subset of the positions marked in the second string. This gives us the set

[7]I have tried to stick to a convention that a diamond ◇ has been used as such a marker.

Table 4: Definitions of the Rule Compiler

Compiler for Optional Rules

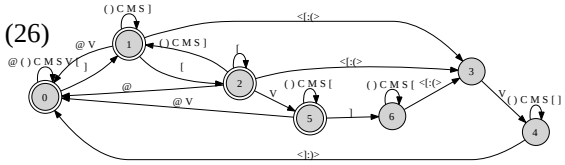
```
def Chg         A:A  - A | A:0 | 0:A ;
def W           Z "<>" Chg Z ;
def Intro       [ A | 0:"<>" ]* ;
def Hide(X)     Intro  .o. X .o. Intro .i ;
def CR(Rule)    Z - Hide(W - Rule) ;
```

Compiler for Obligatory Rules

```
def Chg           A:A - A | A:0 | 0:A ;
def MarkChg(X)    X/"<>" & [A | "<>" Chg ]* ;
def FewerT        ?* [ 0:"<>" ?* ]+ ;
def TooFew(X)     [ MarkChg(CR(X)).1 .o. FewerT .o.
                    MarkChg(CR(X)).1 ].1 ;
def COERCE(Rule) Hide( [ ?* - TooFew(Rule) ]
                     .o. MarkChg(CR(Rule)) ) ;
```

of marked strings that indicate which paths in the first resulting transducer fail applying the rules as often as possible. When these paths are removed from `CR(Rule)`, we obtain a transducer where the rule's application is obligatory whenever there is a choice. For (23b), this transducer is (26).

(26)



6 Evaluation

All linear encodings for autosegmental structure have some limitations. While the strength of the proposed notation is the easiness to link multiple segments to a single autosegment and multiple tones to single segment, it is not perfect concerning the treatment of floating tone. In terms of Kornai's (1995) criteria, it seems to be compositional, computable, and iconic, but not fully invertible because there are such bracketed strings that have no interpretation as a graphical autosegmental representation.

The purpose of the current proposal has been to present just the core ideas of the new representation, not to make universal claims. For example, tones do not always realize (Leben, 2006). The current proposal can be criticized also for other simplifying assumptions about TBUs and tone assignment boundaries.

The original two-level formalism (Koskenniemi, 1983) is difficult to use because it is based on implications and because the rule system may be overconstrained. The new rule formalism seems to address both of these problems, making the rule semantics very much like replace rules in the state-of-the-art finite-state toolkits.

Table 5: # of states in compiled rules

name	Rule	CR	COERCE
AHS	25	7	11
High Lowering	27	7	12
FLD	21	7	13
Plateauing	21	7	16
Meeussen	13	4	7
Tone Shift	235	96	117

Simultaneous compilation of multiple two-level rules has been proposed in (Yli-Jyrä and Koskenniemi, 2006; Yli-Jyrä, 2008a), but this has not been put into practice in full scale, except in one special case (Yli-Jyrä, 2009). The current compilation method compiles, however, all the subrules of the TONE SHIFT simultaneously.

The compilation method is easy to implement and easy to use. In addition, the formalism is flexible because simultaneous, interlinked changes can be described. The only really complex exception discovered so far is the TONE SHIFT rule. Excepting TONE SHIFT, the sizes of rule transducers (Table 5) are quite small.

My resources did not allow me to rewrite a complete tonal description such as (Halme, 2004). I hope, however, that the presented ideas and (really open source) formalism are useful for later efforts.

It would be possible to define rule templates to be used routinely in experimental descriptive efforts. Tentative accounts of various phenomena could then be iteratively tested and debugged with finite-state tools, giving valuable feedback to a descriptive linguist, a resource builder or a theoretic phonologist working on tonal languages. The experimental verification would finally contribute towards the quality and wide applicability of descriptive grammars. The simultaneously applicable rule templates could also facilitate the development of machine learning methods for tonology.

References

Rose-Juliet Anyanwu. 2008. *Fundamentals of Phonetics, Phonology and Tonology*. Number 15 in Schriften zur Afrikanistik - Research in African Studies. Peter Lang.

Lotta Aunio. 2010. Ikoma nominal tone. *Africana Linguistica*, 16:3–30.

Steven Bird and T. Mark Ellison. 1994. One-level phonology: autosegmental representations and

rules as finite automata. *Computational Linguistics*, 20(1).

Steven Bird and Ewan Klein. 1994. Phonological analysis in typed feature systems. *Computational Linguistics*, 20(3):455–491.

Steven Bird. 1995. *Computational Phonology. A constraint-based approach*. Studies in Natural Language Processing. Cambridge University Press.

John Goldsmith. 1973. Tonemic structure. Manuscript, downloaded from `http://hum.uchicago.edu/jagoldsm/Webpage/Courses/HistoryOfPhonology/` in June 2013.

John Goldsmith. 1976. *Autosegmental Phonology*. Ph.D. thesis, MIT.

Riikka Halme. 2004. *A tonal grammar of Kwanyama*. Rüdiger Köppe Verlag, Köln.

Jeffrey Heinz and Cesar Koirala. 2010. Maximum likelihood estimation of feature-based distributions. In *Proceedings of the 11th Meeting of the ACL-SIGMORPHON, ACL 2010*, pages 28–37, Uppsala, Sweden, 15 July.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th EACL: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

William J. Idsardi. 2009. Calculating metrical structure. In Charles Cairns and Eric Raimy, editors, *Contemporary Views on Architecture and Representations in Phonological Theory*, pages 191–211. MIT Press, Cambridge.

Lauri Karttunen. 2006. A finite-state approximation of Optimality Theory: The case of Finnish prosody. In T. Salakoski et al., editor, *FinTAL 2006*, volume 4139 of *LNAI*, pages 4–15, Berlin Heidelberg. Springer-Verla.

Martin Kay. 1987. Nonconcatenative finite-state morphology. In *Proceedings, Third Meeting of the European Chapterof the Association for Computational Linguistics*, pages 2–10.

András Kornai. 1995. *Formal Phonology*. Garland Publishing, New York.

Kimmo Koskenniemi. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

William Leben. 1973. *Suprasegmental phonology*. Ph.D. thesis, MIT.

William R. Leben. 1978. The representation of tone. In Victoria A. Fromkin, editor, *Tone: A Linguistic Survey*. Academic Press, New York.

William R. Leben. 2006. Rethinking autosegmental phonology. In John Mugane et al., editor, *Selected Proceedings of the 35th Annual Conference on African Linguistics*, pages 1–9, Somerville, MA. Cascadilla Proceedings Project.

Jackson Muhirwe. 2010. Morphological analysis of tone marked kinyarwanda text. In Anssi Yli-Jyr, Andrs Kornai, Jacques Sakarovitch, and Bruce Watson, editors, *Finite-State Methods and Natural Language Processing*, volume 6062 of *Lecture Notes in Computer Science*, pages 48–55. Springer Berlin Heidelberg.

Alan Prince and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishing.

Bruce Wiebe. 1992. Modelling autosegmental phonology with multitape finite state transducers. Master's thesis, Simon Fraser University.

Edwin S. Williams. 1976. Underlying tone in Margi and Igbo. *Linguistic Inquiry*, 7:463–484.

Anssi Yli-Jyrä and Kimmo Koskenniemi. 2006. Compiling generalized two-level rules and grammars. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Advances in Natural Language Processing*, volume 4139 of *Lecture Notes in Computer Science*, pages 174–185. Springer Berlin Heidelberg.

Anssi Yli-Jyrä. 2008a. Applications of diamonded double negation. In T. Hanneforth and K-M. Würzner, editors, *Finite-State Methods and Natural Language Processing, 6th International Workshop, FSMNL-2007, Potsdam, Germany, September 14–16, Revised Papers*, pages 6–30. Potsdam University Press, Potsdam.

Anssi Yli-Jyrä. 2008b. Transducers from parallel replacement rules and modes with generalized lenient composition. In T. Hanneforth and K-M. Würzner, editors, *Finite-State Methods and Natural Language Processing, 6th International Workshop, FSMNL-2007, Potsdam, Germany, September 14–16, Revised Papers*, pages 197–212, Potsdam. Potsdam University Press.

Anssi Yli-Jyrä. 2009. An efficient double complementation algorithm for superposition-based finite-state morphology. In *Proceedings of NODALIDA 2009*, pages 206–213.