Three Equivalent Codes for Autosegmental Representations

Anssi Yli-Jyrä
Department of Modern Languages, University of Helsinki, Finland

Abstract

Autosegmental representations are typically bipartite graphs that describe how a tonal string aligns with a segmental string. Recently, a simple bracket-based encoding (Yli-Jyrä 2013) has enabled applying graph rewriting rules to a class of encoded bipartite graphs used in tonology. The resulting string transductions are compatible with finite state approaches to computational morphology. To manipulate a bigger class of graphs, the current work presents three new codes for them: (i) the first code is used to visualize the autosegmental representations, (ii) the second code generalizes the prior bracket-based linearization and (iii) the third code has bijections to the other two codes. Their code words correspond to *embracements* that are the smallest subgraphs from which the graphs can be concatenatively generated without adding new edges. These infinite codes demonstrate, finally, the existence of a compositional coding function (Kornai 1995) for the class of *inertial* autosegmental graphs where *the unlinked units do not float without the force of graph rewrites.*

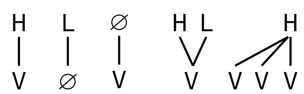
1 Introduction

There are few other phonological phenomena as complex as the Bantu tone (Nurse and Philippson, 2003) that is exhibited by some 500 Bantu dialects spoken in Africa. Computational modeling of tone is crucial for speech technology of these and many other tonal languages because phonological tone contributes to lexical and grammatical disambiguation of speech, especially in noisy environments. But the inherent complexity of tone and the lack of data makes it difficult to extend *lexical transducers* (Beesley and Karttunen, 2003) needed in speech processing into *tonal lexical transducers* (Hurskainen, 2009; Muhirwe, 2010) without an adequately implemented theory of tone.

1.1 Autosegmental Phonology

Autosegmental Phonology (Goldsmith, 1976) is widely recognized as the most important phonological theory when it comes to description of tone. The theory has made a long-standing impact on phonology and speech technology by formalizing the ideas such as the *Obligatory Contour Principle (OCP)* that bans adjacent copies of units, the *independence of the melody* and the *decomposition of contour tones as level tones.* Its multi-tiered representation has crucially influenced more complex phonological theories such as Feature Geometry (Clements, 1985), Optimality Theory (Prince and Smolensky, 2004; McCarthy, 2011), Optimal Domains Theory (Cassimjee and Kisseberth, 1998) and Autosegmental-Metrical Theory (Ladd, 2008). We believe that a computational implementation of Autosegmental Phonology will have far-reaching consequences, paving the way for fuller implementation of theories that involve multiple feature tiers or a class of alignment structures beyond phonology.

Autosegmental Phonology states that a tone is not a feature inside a phonemic segment, but rather a segment in its own right – thus an *auto*segment. The phonological representation consists of aligned tiers of strings: the autosegmental tonemes lying in one tier are associated with segmental phonemes in the other tier. For example, one possible way to associate the tone string HLHLH with the vowel string VVVVVV is[1]:



(1)

The notion of autosegments formalizes the five important characteristics (Yip, 2002) of the tone-segment relationship during the derivation of the surface forms:

1. **"mobility":** the alignment between tones and segments is not fixed by the lexicon, but can change during the derivation.

[1] We naively assume that the *tone bearing units (TBUs)* in the segmental string are vowels (V), although they could be also moras (μ) or syllables (σ).

2. **“stability of tone”:** a tone can be temporarily unattached to a segment (∅ = none).
3. **“toneless segments”:** a segment can be temporarily unattached to a tone.
4. **“one-to-many”:** a tone can be associated with multiple segments.
5. **“many-to-one”:** many tones can be associated with a single segment.

1.2 Prior Computational Approaches

A few computational implementation approaches for autosegmental representation have been proposed. Firstly, there are interesting implementation efforts that have resulted into feature-rich systems such as the Delta programming language (Hertz, 1990), AMAR (Albro, 1994) and Tone Pars (Black, 1997). These systems are powerful but do not seem to be compatible with finite state transducers. Secondly, there are approaches that are based on finite automata and transducers or equivalent mechanisms.

The finite state approaches can be divided into two main categories depending on the role they assign to the OCP:

1. Strict/immediate approaches (Bird and Klein, 1990; Bird and Ellison, 1994; Bird, 1995; Carson-Berndsen, 1998; Jardine, 2013) hardwire the OCP principle into their mathematical formalization of autosegments. This approach is supported by Leben (1973).
2. Lenient/lazy approaches (Kornai, 1995; Wiebe, 1992; Yli-Jyrä, 2013) see that the OCP can sometimes be violated and must be actively maintained by the rules (Odden, 1986). This approach is supported by some existing theoretical work (Goldsmith, 1976; Odden, 1986; Prince and Smolensky, 2004) and field linguistic descriptions (Halme, 2004).

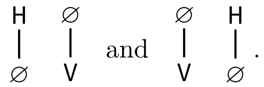
Both approaches have their merits. They are also based on finite state machines and can be emulated by one another. The strict formalization supports easier grammar inference because it focuses on the surface representation. In contrast, the lenient formalization of the OCP tolerates a wider range of rule systems. Therefore, the lenient OCP approach seems to us more relevant considering the possible needs during the practical construction of lexical transducers from existing derivational descriptions. The relevant contributions towards its implementation can be summarized as follows:

- Wiebe (1992) and Kornai (1995) model the alignment of tone via the moves of multi-tape automata and discuss the possibility of corresponding linear codes.
- Yli-Jyrä (2013) encodes and rewrites autosegmental representations via simple bracketed strings. This encoding is practically implemented and compatible with the standard technology for building lexical transducers.

1.3 The Features of Simple Bracketing

In the bracket-based encoding (Yli-Jyrä, 2013), the edge of each tone is indicated with a corresponding bracket. For example, the representation (1) reduces to string [V]()V[V)[VVV]. When we replace symbols ()[] with $\overline{\text{H}}\text{H}\,\overline{\text{L}}\text{L}$, respectively, this string reduces to $\overline{\text{L}}\text{VL}\,\overline{\text{H}}\text{H}\,\text{V}\,\overline{\text{H}}\text{VL}\,\overline{\text{H}}\text{VVVH}$.

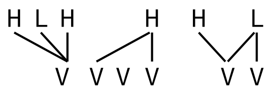
The bracket-based encoding has some known design features and limitations. According to the Inertia property[2], the encoding distinguishes between the representations



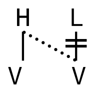
since the order of tone-segment associations and unlinked elements determine the linearization. By ordering the unlinked elements, especially under concatenation, Inertia makes more distinctions than what is usually linguistically motivated. The resulting spurious ambiguity can, however, eliminated by deterministic choices and rule conditions that match different choices. On the other hand, the additional structure due to Inertia contributes towards the existence of a coding morphism.

Our attention is in the actual limitations of the encoding scheme:

- The encoding is not fully treated as a formal code that maps a (possibly infinite) set of component graphs into code words. Therefore, the negative result of the non-existence of an *optimal coding function* (Kornai, 1995) remains.
- The encoding does not capture some tone configurations of autosegmental bipartite graphs: complex contour tones, skipping of unlinked vertices and zigzag chaining:



- Associations and respective changes must be hand-encoded, while the original formalism (Goldsmith, 1976) is graphical and it indicates insertions (⁝) and removals (∦) intuitively by graphical signs:



[2] Introduced in Yli-Jyrä (2015).

1.4 The Results of This Paper

This paper focuses to linearization of two-tiered noncrossing representations, approaching the linearization problem via the coding theory (Berstel et al., 2010).

The first main result (Section 3) of this approach is that an **optimal coding function** (Kornai, 1995) for inertial autosegmental graphs exists under an appropriately defined *infinite set of generators*.

The second major result is that a *serial code with an optimal coding function can be visual* too (Section 4). Our **visual code** is easy to read and write, which is an important feature during system development.

The third major result is that the previously presented bracket encoding extends to all non-crossing link configurations via the **symmetric bracket code** that supports complex contours, skipping and chaining (Section 5).

Finally, we show that the visual code and the symmetric bracket code are related via bijections with a **Polish code** that is simple and efficient, but the least visual (Section 6). The visual and the Polish codes can be extended to represent simple rewriting rules.

2 Definitions

2.1 Autosegmental Graphs

Different axiomatizations for autosegmental graphs have been proposed since Goldsmith (1976). The tone-segment associations are formalized as a graph whose vertices are (auto)segments. The sets of vertices are ordered since they correspond to tonal and segmental strings. The associating edges between them are ordered due to the (NON-CROSSING CONSTRAINT, NCC). The NCC constraint is common assumption in autosegmental theory, although it has been contested in prosodic morphology (Bagemihl, 1989; McCarthy and Prince, 1996). A vertex without incident edges is called an *isolate*. The set of all isolates is ordered under the INERTIA property.

Definition 1. *An* autosegmental graph *is*

- *a bipartite graph $G = (T \cup V, E, \leq_T, \leq_V, \leq_E)$ with vertices $T \cup V$ and undirected edges $E \subseteq T \times V$,*
- *with total order $\leq_T$ for T and $\leq_V$ for V,*
- *with total order $\leq_E$ of edges E satisfying $(v, v') \leq (u, u') \leftrightarrow v \leq u \wedge v' \leq u'$ for all $(v, v'), (u, u') \in E$.*

If there is a total order $\leq_I$ over $(T \cup V) - \{e \mid (e, f) \in E\}$, $G = (T \cup V, E, \leq_T, \leq_V, \leq_E, \leq_I)$ is an autosegmental graph with INERTIA.

Concatenation of disjoint a.s. graphs is defined by extending the $\leq$-relations in a natural way, by concatenating the ordered sets of tones, vowels, edges and isolates.

2.2 Coding Functions

Kornai (1995) formalized computational implementation of autosegmental graphs as the problem of finding a *coding function* β such that for any autosegmental graph G (called a *bistring* by Kornai), its image $\beta(G)$ is a string over a finite alphabet A.

There are infinitely many possible coding functions, but Kornai narrows the task with four properties characterizing *an optimal coding function* $\hat{\beta}$:

1. $\hat{\beta}$ is **computable** e.g. by a Turing Machine; ideally, it is computable with a finite state transducer.
2. $\hat{\beta}$ is **invertible** i.e. at least injective; ideally, this means that function $\hat{\beta}$ is even bijective.
3. $\hat{\beta}$ is **iconic** i.e. analogical rather than cryptic; ideally, a local change in G corresponds to a local change in $\hat{\beta}(G)$.
4. $\hat{\beta}$ is **compositional** i.e. it is a morphism that respects concatenation: $\hat{\beta}(GG') = \hat{\beta}(G)\hat{\beta}(G')$.[3]

It is impossible to construct an ideal coding function for autosegmental graphs in general (Kornai, 1995). The lack of compositionality holds even when these graphs are edgeless (Wiebe, 1992). This relates to the fact that the Cartesian product $A^* \times B^*$ is not equidivisible and thus not a free monoid (Sakarovitch, 2009).

2.3 Towards a Monoid Morphism

Although there is no compositional coding function for all autosegmental graphs (Wiebe, 1992), we will now exploit coding theory (Berstel et al., 2010) to find a class of autosegmental graphs for which a compositional coding function exists.

Let A be an alphabet. A subset $X \subseteq A^*$ is a *code* over A if every word $w \in X^*$ is written *uniquely* as a product of words in X. In other words, for $x_1, \ldots, x_n, y_1 \ldots y_m \in X$, the condition $x_1 \ldots x_n = y_1 \ldots y_m$ implies $n = m$ and $x_i = y_i$ for all l $i \in \{1, .., n\}$.

If the concatenation is defined over disjoint graphs and it does not add new edges, there must be a code element for every connected graph. Since the set of connected graphs is infinite, a block code is not possible. Instead, the code must be *infinite*. For example, let $S \subseteq A^+$ be a set of *semaphores*. The *maximal semaphore code* $X = A^*S - A^*SA^+$ contains words that end with a semaphore in S but

[3] Beware the terminology: β can be compositional without being *composable* (Berstel et al., 2010).

do not otherwise contain a semaphore. This code is infinite at least when $S \subset A$. Any subset of a code is also a code. For other classes of infinite codes, the reader is referred to (Berstel et al., 2010).

A morphism $\beta : H^* \to A^*$ from a monoid $(H^*, \cdot, \epsilon)$ into a monoid $(A^*, \cdot, \epsilon)$ is a *coding morphism* for a code $X \subseteq A^*$ if

1. β is injective and, thus, preserves the distinctiveness of the elements of H

2. $X = \beta(H)$, i.e., every code word corresponds to a letter in H. (Berstel et al., 2010)

Theorem 1. *Let $\beta : H^* \to A^*$ be a coding morphism for a code $X \subseteq A^*$. Then β is an invertible and compositional coding function (Kornai, 1995).*

Proof. Every morphism is compositional since the condition $\beta(hh') = \beta(h)\beta(h')$ for all $h, h' \in H^*$ is a part of the definition of morphisms. Moreover, invertibility follows from the definition of a coding morphism. □

2.4 The Approach

Our current code theoretic approach can be summarized as follows:

1. **No free floating**. Since the graphs must correspond to uniquely tokenizable code words, we will restrict ourselves to the autosegmental graphs with INERTIA.

2. **Infinite source alphabet and code**. Theorem 1 and the previous negative results guide us to reject the prior assumption about a finite code (Kornai, 1995) and look for an infinite code X and an infinite source alphabet H consisting of certain kinds of graphs. The resulting approach differs vastly from approaches that enforce a finite source alphabet and the OCP (e.g. Bird and Ellison 1994).

3. **Finite channel alphabet**. Defining morphisms from an infinitely generated monoid may seem scary, as formal languages are usually subsets of finitely generated free monoids. However, after the graphs are encoded by a code X^* that is subset of a finitely generated monoid A^*, they can be manipulated indirectly via finite state relations over subsets of A^*.

3 Codability of Graphs

We will now choose a source alphabet H in such a way that it generates a free monoid $(H^*, \cdot, \epsilon)$ and coincides with our autosegmental graphs.

3.1 Non-trivial Connected Components

Theorem 2. *All connected components of autosegmental graphs are trees.*

Proof. Assume that there is a connected autosegmental graph that is not a tree. Such a bipartite graph must contain a cycle that involves at least four vertices and there is a crossing edge; contradiction. □

Define the relation $\leq_S$ over subgraphs C and C' of a graph G in such a way that $C \leq_S C'$ if $C = C'$ or every edge in C is before every edge in C'. The relation $\leq_S$ is reflexive, transitive and antisymmetric, thus a partial order.

Theorem 3. *The $\leq$-order of nontrivial (containing at least two vertices) connected components of an autosegmental graph is a total order.*

Proof. Let C and C' be nontrivial connected components for which $C \not\leq_S C'$ and $C' \not\leq_S C$. Then there are edges $(s, s'), (t, t') \in C$, $(u, u') \in C'$ or edges $(s, s'), (t, t') \in C'$, $(u, u') \in C'$ such that $(s, s') <_E (u, u') <_E (t, t')$. Since s and t are connected and there are no crossing edges, they are connected via u or u' (or both). This means, however, that C and C' are connected and $C = C'$; contradiction. Thus, the order $\leq_S$ is total. □

3.2 Embracements

An isolate vertex y is *embraced* by a connected component C if C contains vertices x and z such that $x \leq_T y \leq_T z$ or $x \leq_V y \leq_V z$. We extend every connected component C of an autosegmental graph into a subgraph C' called an *embracement* by adding to C' all isolates embraced by C.

Corollary 1. *Every autosegmental graph consists of a totally ordered set of embracements.*

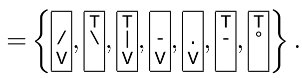
Thus, we obtain a *complete set of generators* for all autosegmental graphs by expanding the connected components with embraced isolates. This infinite set of generators H is such that an optimal coding function $\hat{\beta} : H^* \to A^*$ is possible.

4 New Codes for the Graphs

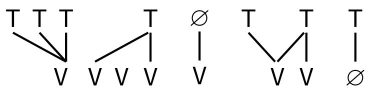
4.1 The Visual Code

The visual code is defined over the alphabet

$$A = \{(/\mathrm{V}\ ), (\backslash\ \mathrm{T}), (|\mathrm{VT}), (-\mathrm{V}\ ), (.\mathrm{V}\ ), (-\ \mathrm{T}), (^\circ\ \mathrm{T})\}$$



The idea is that autosegmental graphs such as


(2)

are encoded as visually similar strings:

$$\boxed{T\backslash}\boxed{T\backslash}\boxed{T|V}\ \boxed{/V}\boxed{-V}\boxed{T|V}\ \boxed{.V}\ \boxed{T\backslash}\boxed{/V}\boxed{T|V}\ \boxed{T°}.$$

The code is infinite and has a value for every possible embracement. The code word for each embracement ends with one of the semaphores $S = \{(.\mathsf{V}), (°\ \mathsf{T}), (|\mathsf{VT})\}$. Corresponding to this, we designate one of the rightmost vertices as the root for any non-trivial tree in the embracement. The code word for an embracement is constructed leftwards from this vertex as explained in Table 1. The resulting set of code words is given by regular expression[4]

$$X = \left\{\boxed{.V} \bigcup \boxed{T°}\right\} \cup \left\{\boxed{/V}\ \boxed{-V}^{*} \bigcup \boxed{T\backslash}\ \boxed{T-}^{*}\right\}^{*} \boxed{T|V}.$$

This set is recognized by the automaton in Fig. 1.

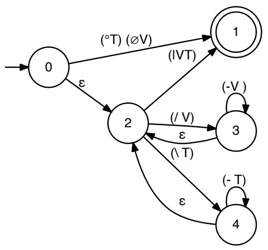


Figure 1: Recognizer for the visual code

Theorem 4. *The language X is a code.*

Proof. The set X is a subset of the maximal semaphore code $A^*S - A^*SA^+$ with semaphores S. Thus, X is a code. □

4.2 The Symmetric Bracket Code

The alphabet of the symmetric bracket code is $B = \{[,], \bar{\mathsf{V}}, \mathsf{V}, °\}$. As the vowels and tones reduce to paired symbols i.e. brackets, *both* vertex types are treated symmetrically as autosegments. For example, (2) is encoded with the symmetric bracket code as

$$\bar{\mathsf{V}}[\,][\,][\mathsf{V}]\ \ \bar{\mathsf{V}}[\mathsf{V}\bar{\mathsf{V}}°\mathsf{V}\bar{\mathsf{V}}\mathsf{V}]\ \ \bar{\mathsf{V}}\mathsf{V}\ \ \bar{\mathsf{V}}[\,][\mathsf{V}\bar{\mathsf{V}}\mathsf{V}]\ \ [\,].$$

The code word for each embracement ends with a closing tone bracket ($]$) or a closing vowel bracket (V). The code word for an embracement is constructed as explained in Table 2. The resulting set of code words is

$$Z = \left\{\bar{\mathsf{V}}\mathsf{V} \bigcup [\,]\right\} \bigcup$$
$$\bar{\mathsf{V}}[\left\{\ ]\left\{[°]\right\}^{*}[\ \bigcup\ \mathsf{V}\left\{\bar{\mathsf{V}}°\mathsf{V}\right\}^{*}\bar{\mathsf{V}}\ \right\}^{*}\mathsf{V}].$$

[4]Here the regular expression operators are: concatenation, union (∪), star (*), kleene plus ("+"), difference (−), and grouping ({}).

Table 1: Encoding a graph with the visual code

$\boxed{.V}, \boxed{T°}$	Non-embraced isolated vertices are indicated by respective symbols that are ordered according to the order of the isolates.
$\boxed{T\|V}$	The rightmost edge of each non-trivial tree is indicated by a bar.
$\boxed{/V}\ldots\boxed{/V}\boxed{T\|V}$	Additional vowels can be linked to the tone of the rightmost vertice (the natural limit is, however, three tones).
$\boxed{T\backslash}\ldots\boxed{T\backslash}\boxed{T\|V}$	Additional tones can be linked to the vowel of the rightmost vertice.
$\boxed{T\backslash}\ldots\boxed{T\backslash}\boxed{/V}\ldots\boxed{/V}\boxed{T\|V}$ (shared vertex↑)	Any shared vertex starts a new subtree.
$\ldots\boxed{/V}\boxed{T\backslash}\ldots\boxed{T\backslash}\boxed{/V}\ldots\boxed{/V}\boxed{T\backslash}\ldots$	There is no limit for the depth of chained subtrees.
$\boxed{/V}\boxed{-V}\boxed{T\|V}\ \boxed{T\backslash}\boxed{T-}\boxed{T\|V}$	Embraced isolates are indicated with a dash.

Theorem 5. *The language Z is a code.*

Proof. The finite automaton in Figure 2 recognizes the language Z. Although the drawn automaton is not deterministic, it is easy to verify that it recognizes a prefix-free stringset — thus a code (Berstel et al., 2010). □

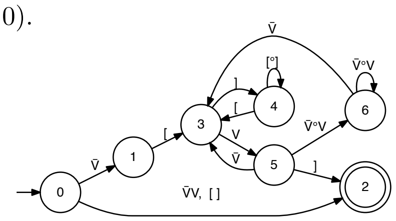


Figure 2: Recognizer for the bracket code

4.3 The Polish Code

The reverse Polish notation due to Jan Łukasiewicz (Hamblin, 1962) is a parenthesis/bracket-free notation for mathematical expressions. In this notation, every operator follows all its operands. The notation can be generalized to cases where every operand is operated twice.

Our Polish code is defined over the alphabet $D = \{|, °, \mathsf{V}, \mathsf{T}\}$. For example, (2) is encoded as

$$|\mathsf{T}|\mathsf{T}|\mathsf{VT}\ \ |\mathsf{V}°\mathsf{V}|\mathsf{VT}\ \ \mathsf{V}\ \ |\mathsf{T}|\mathsf{V}|\mathsf{VT}\ \ \mathsf{T}. \qquad (3)$$

Every edge ($|$) acts twice as an operand: for two kinds of variadic operators, a tone (T) and a vowel (V). One of these operators must be applied immediately to each edge, but the other operator is later applied to all edges that have not been operated on. If an operator would otherwise have

Table 2: Encoding with the bracket code

$[\,] = \boxed{\substack{\mathsf{T}\\ \circ}}$, $\overline{\mathsf{V}}\mathsf{V} = \boxed{\substack{\cdot\\ \mathsf{V}}}$	Non-embraced isolates correspond to simple brackets.
$\overline{\mathsf{V}}[\mathsf{V}] = \boxed{\substack{\mathsf{T}\\ \vert\\ \mathsf{V}}}$	A one-edge tree is indicated with tone brackets that overlap with segment brackets.
$\overline{\mathsf{V}}[\mathsf{V}\overline{\mathsf{V}}\mathsf{V}\cdots\overline{\mathsf{V}}\mathsf{V}]$	Additional vowels can be linked to a tone after the first vowel.
$\overline{\mathsf{V}}[\,]\cdots[\,][\mathsf{V}]$	Additional tones can be linked to a vowel before the last tone.
$\overline{\mathsf{V}}\underbrace{[\mathsf{V}\overline{\mathsf{V}}\mathsf{V}\cdots\ \overbrace{\overline{\mathsf{V}}][\,]\cdots[}^{\text{daughter's tree}}}_{\text{gr.dtr's tree}}\underbrace{\mathsf{V}\overline{\mathsf{V}}\cdots\mathsf{V}]}_{\text{topmost tree}}$	Chained trees share vertices via interlocking.
$\cdots\overline{\mathsf{V}}[\mathsf{V}\overline{\mathsf{V}}]\cdots[\mathsf{V}\overline{\mathsf{V}}][\mathsf{V}\cdots]$	There is no limit for the depth of chains.
$\overline{\mathsf{V}}[\mathsf{V}\overline{\mathsf{V}}{}^\circ\mathsf{V}\overline{\mathsf{V}}\mathsf{V}]$ $\overline{\mathsf{V}}[\,][{}^\circ][\mathsf{V}]$	Embraced isolates are indicated with a small circle.

some operands available, it can be forced to apply to *no* operands (edges) through a °-prefix. In order to avoid spurious ambiguity, sequences $|\mathsf{VT}$, $|\mathsf{VT}\cdots\mathsf{T}$, $|\mathsf{VTV}\cdots\mathsf{V}$ are equated with their non-normalized equivalents $|\mathsf{TV}$, $|\mathsf{T}\cdots\mathsf{TV}$, $|\mathsf{VV}\cdots\mathsf{VT}$ that will be excluded from the code. The code word for an embracement is constructed as explained in Table 3. The resulting set of code words is

$$Y = |\left\{\mathsf{V}\left\{{}^\circ\mathsf{V}\right\}^* | \bigcup \mathsf{T}\left\{{}^\circ\mathsf{T}\right\}^* |\right\}^* \mathsf{VT} \bigcup \left\{\mathsf{V},\mathsf{T}\right\}.$$

These code words are recognized by the finite automaton in Figure 3.

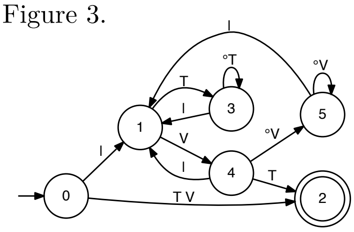


Figure 3: Recognizer for the Polish code

Theorem 6. *The language Y is a code.*

Proof. The language Y is a *prefix code*, thus, a code: $Y \cap YD^+ = \emptyset$ (Berstel et al., 2010). □

5 Equivalences

In this section, we show that the autosegmental graphs with Inertia have linear time coding functions to the three codes presented above. In addition, we show that the bijections between the codes are finite-state computable. This results into equivalences in Figure 4.

Table 3: Encoding a graph with the Polish code

T, V	Non-embraced isolates are nullary operators.
$\vert\mathsf{VT}$	A one-edge tree consists of an edge and two vertices.
$\vert\mathsf{V}\cdots\vert\mathsf{V}\vert\mathsf{VT}$	Additional vowels can be linked.
$\vert\mathsf{T}\cdots\vert\mathsf{T}\vert\mathsf{VT}$	Additional tones can be linked.
$\vert\mathsf{T}\vert\mathsf{V}\vert\mathsf{V}\vert\mathsf{T}\vert\mathsf{T}\vert\mathsf{V}\vert\mathsf{VT}$	Trees can be chained.
$\vert\mathsf{V}{}^\circ\mathsf{V}\vert\mathsf{VT}$ $\vert\mathsf{T}{}^\circ\mathsf{T}\vert\mathsf{VT}$	Embraced isolates are nullary operators.

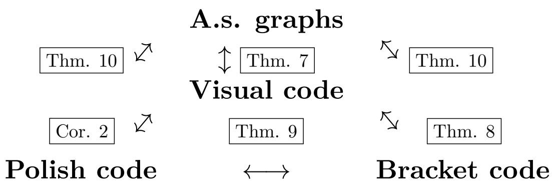


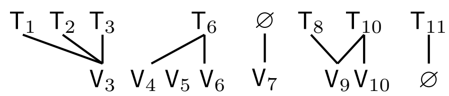
Figure 4: The equivalence results

Theorem 7. *There is a linear time bijection between the visual code and the set of possible embracements in autosegmental graphs.*

Proof. Let $G = (T \cup V, E, \leq_T, \leq_V, \leq_E, \leq_I)$ be an autosegmental graph with Inertia. Compute a partial order $\leq_F$ over $T \cup V$ in such a way that

$$\begin{aligned} y <_F z \quad & \text{if and only if } \exists (y,z),(x,z) \in E \text{ s.t.} \\ & y <_T x \text{ or } y <_V x \\ y =_F z \quad & \text{if and only if } \exists (y,z) \in E \text{ s.t} \\ & \forall (x,z),(y,v) \in E \text{ we have} \\ & (x,z) \leq_E (y,z) \text{ and } (y,z) \leq_E (y,z). \end{aligned}$$

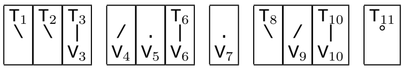
Let $G' = (T \cup V, \leq_T \cup \leq_V \cup \leq_I \cup \leq_F)$ be a directed graph. If this graph contains cycles they are between $=_F$ equivalent vertices. We eliminate each cycle by unifying the vertices in it. The resulting acyclic graph is sorted topologically to obtain a totally ordered graph such as:



Using the total order over vertices, we encode each position with an element of the alphabet A of the visual code. In total, the encoding is computed in deterministic linear time.

Conversely, let $w \in X^*$ be a visual code string. We construct an autosegmental graph $G = (T \cup V, E, \leq_T, \leq_V, \leq_E, \leq_I)$ from it: The boxes are numbered rightward by positive integers. These numbers give indices to the vertices. Then we extract, from w, the edges by complementing each partial edge with the closes missing vertex to the right. Then the total orders $\leq_T, \leq_V, \leq_E, \leq_I$ are extracted in the obvious way by a finite number of scans through the string.

For example, the visual code string



gives rise to the autosegmental graph $G = (T \cup V, E, \leq_T, \leq_V, \leq_E, \leq_I)$ with

- the $\leq_T$-ordered set of tone vertices $T = \langle \mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_6, \mathsf{T}_8, \mathsf{T}_{10}, \mathsf{T}_{11} \rangle$
- the $\leq_V$-ordered set of vowel vertices $V = \langle \mathsf{V}_3, \mathsf{V}_4, \mathsf{V}_5, \mathsf{V}_6, \mathsf{V}_7, \mathsf{V}_9, \mathsf{V}_{10} \rangle$,
- the $\leq_E$-ordered set of edges $E = \langle (\mathsf{T}_1,\mathsf{V}_3), (\mathsf{T}_2,\mathsf{V}_3), (\mathsf{T}_3,\mathsf{V}_3), (\mathsf{T}_6,\mathsf{V}_4), (\mathsf{T}_6,\mathsf{V}_6), (\mathsf{T}_8,\mathsf{V}_9), (\mathsf{T}_{10},\mathsf{V}_9), (\mathsf{T}_{10},\mathsf{V}_{10}) \rangle$,
- the $\leq_I$-order over isolates: $\mathsf{V}_5 < \mathsf{V}_7 < \mathsf{T}_{11}$. □

Theorem 8. *There is a finite state bijection between the Polish code and the visual code.*

Proof. Figure 5 shows a finite state transducer mapping the language Y^* to language X^*. It is easy to verify that both the transducer and its inverse are functions. In addition, it is possible to verify mechanically that the preimage and the image of this function are exactly Y^* and X^*, respectively. Thus, it defines a bijection.

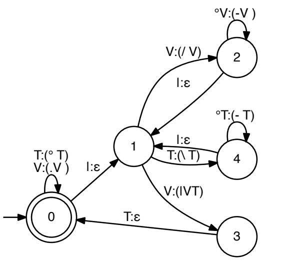


Figure 5: $Y^* \leftrightarrow X^*$

The transducer in Figure 5 is constructed with the 2-tape regular expression[5]:

$$\left\{ \begin{array}{c} \mathsf{|}{:}\epsilon \left\{ \begin{array}{c} \mathsf{T}{:}(\backslash\ \mathsf{T})\ \ {}^\circ\mathsf{T}{:}(\text{-}\ \mathsf{T})^*\ \mathsf{|}{:}\epsilon \\ \cup \\ \mathsf{V}{:}(/\ \mathsf{V})\ \ {}^\circ\mathsf{V}{:}(\text{-}\mathsf{V}\ \ )^*\ \mathsf{|}{:}\epsilon \end{array} \right\} ^* \mathsf{V}{:}(\,\mathsf{|VT})\ \mathsf{T}{:}\epsilon \\ \cup \\ \mathsf{T}{:}(^\circ\ \mathsf{T}) \cup \mathsf{V}{:}(\,.\ \ \mathsf{V}) \end{array} \right\}^*$$

□

Theorem 9. *There is a finite state bijection between the Polish code and the symmetric bracket code.*

Proof. There is a finite state transducer (Figure 6) mapping strings Y^* to strings Z^*. The preimage and the image of this function are Y^* and Z^*, respectively. It is easy to verify that both the transducer and its inverse are total functions. Thus, it defines a bijection.

[5]The additional operators are: cross product (:), and composition (∘)

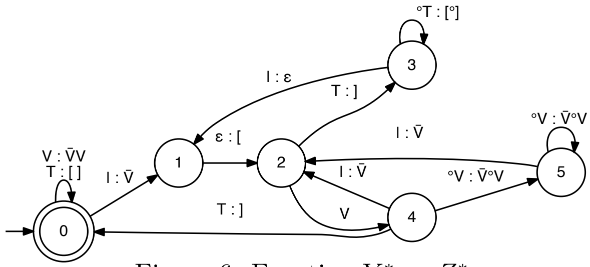


Figure 6: Function $Y^* \leftrightarrow Z^*$.

The transducer in Figure 6 is constructed by the 2-tape regular expression

$$Y^* \circ \left\{ \begin{array}{c} (B\backslash\{\mathsf{|} \cup \mathsf{T} \cup {}^\circ\}) \cup \mathsf{T}{:}[\,] \cup {}^\circ\mathsf{T}{:}[\,^\circ\,] \cup {}^\circ\mathsf{V} \\ \cup \\ \mathsf{|}\ \{\ \epsilon{:}[\ \}\ \{B\backslash\mathsf{T}\}^*\ \{\ \mathsf{T}{:}]\ \} \end{array} \right\}^*$$

$$\circ \left\{ \begin{array}{c} (B\backslash\{\mathsf{|} \cup \mathsf{V}\}) \cup \mathsf{V}{:}\bar{\mathsf{V}}\mathsf{V} \cup {}^\circ\mathsf{V}{:}\bar{\mathsf{V}}{}^\circ\mathsf{V} \\ \cup \\ \{\mathsf{|}{:}\bar{\mathsf{V}}\}\ \{B\backslash\mathsf{V}\}^*\ \{\mathsf{V}{:}\mathsf{V}\} \end{array} \right\}^*$$

$$\circ \left\{ (B\backslash\mathsf{|}) \cup \mathsf{|}{:}\epsilon \right\}^* .$$

that implements the following intuition: (1) start with Y^*; (2) add tone brackets; (3) add vowel brackets; (4) remove the remaining |-symbols. When these steps are applied to input string (3), we obtain the following derivation:

$$\mathsf{|T^\circ T|VT\ \ |V^\circ V|VT\ \ V\ \ |T|V|VT\ \ T} \Rightarrow$$
$$\mathsf{|[\,][^\circ]|[V]\ \ |[V^\circ V|V]\ \ V\ \ |[\,]|[V|V]\ \ [\,]} \Rightarrow$$
$$\mathsf{\bar{V}[\,][^\circ]|[V]\ \ \bar{V}[V\bar{V}^\circ V\bar{V}V]\ \ \bar{V}V\ \ \bar{V}[\,]|[V\bar{V}V]\ \ [\,]} \Rightarrow$$
$$\mathsf{\bar{V}[\,][^\circ][V]\ \ \bar{V}[V\bar{V}^\circ V\bar{V}V]\ \ \bar{V}V\ \ \bar{V}[\,][V\bar{V}V]\ \ [\,]}.$$

□

Corollary 2. *There is a finite state bijection between the symmetric bracket code and the visual code.*

Theorem 10. *The direct constructions of the symmetric bracket encoding and the Polish encoding from any autosegmental graph with* Inertia *produces the same encoded strings as obtained indirectly via the visual code.*

Proof. Clearly the same set of vertices and edges are maintained by the alternative constructions. We only need to show that also their order relations are maintained regardless of the construction. For the lack of space, this proof is given only intuitively:

- The bijection $Y^* \leftrightarrow X^*$ implemented by transducer in Figure 5 retains the total order of tones, vowels, edges and isolates. The same order is respected by the construction given intuitively in Table 3.
- The bijection $Y^* \leftrightarrow Z^*$ implemented by transducer in Figure 6 retains the total order of tones, vowels, edges and isolates. The same order is respected by the construction given intuitively in Table 2.

Thus, the considered indirect encoding methods are equivalent to the direct ones that are now given only intuitively. □

6 Concluding Remarks

6.1 Immediate Extensions

The current presentation has left out the discussion of the finer distinctions between the vertexes. This has kept the discussion focused on the essence of the unlabeled autosegmental graphs. The refined discussion on vertex labels and the possible OCP principle is, thus, postponed. Nevertheless, the code must be extended to meet various practical requirements. Fortunately, some extensions are immediate or relatively easy as they do not require redesign of the codes.

- **Adding More Vertex Types.** In more elaborated uses of autosegmental graphs, there are several tone types (such as H and L) and distinct vowels. In our approach, such distinctions can be encoded simply by expanding the alphabet with labeled vertexes. In the symmetric bracket code, we can follow the example of (Yli-Jyrä, 2013) and have different sets of brackets for different level tones although it is not necessary to have both sides of the brackets labeled.
- **Adding More Segments.** In addition to labeled T and V vertices, we often need to add other phonemic segments and lexical-morphological features as vertices into the graphs. The set of vowels V can be easily extended to contain all segmental phonemes and morphological features as isolated vertices. Such additional material increases the effect of Inertia, but does not prevent migration of floating elements under appropriate rules.
- **Adding Metrical Structure.** We have naively assumed that vowels correspond to tone bearing units. However, there are languages where the TBUs are moras or syllables. Extensions for such descriptions are feasible but not elaborated in this paper.
- **Adding More Edge Types.** The rule formalism of autosegmental phonology is graphical and uses graphical clues (dotted lines and overstriking) to indicate how the rule rewrites an autosegmental representation. Typical rules can be visualized without separated input and output graphs using *fading-in* edges ⁝, ⋰, ⋱ and *fading-out* edges ǂ, ≯, ≮. Such edges could be used in the visual code as well as in the Polish code. For example, rules

can be written, respectively, as

6.2 Encoded Graph Rewriting

The encoded graphs are strings and they can be closed under such operations as union, concatenation, Kleene star, quotients, and difference. *Encoded graph rewriting* via string transducers seems to be a problem domain where many new results are still to be uncovered.

In Yli-Jyrä (2013; 2015), simpler autosegmental graphs are encoded and rewritten by string transducers, but the current work goes deeper in the code theoretic analysis of the actual encoding functions. Unfortunately, the space does not let us to include an updated discussion of graph rewriting via our new codes for autosegmental graphs.

6.3 The Conclusion of the Results

The current work links the coding theory to linearization of autosegmental graphs and obtains the following results:

- There is a compositional coding function for the infinitely generated set of autosegmental graphs (Thm. 1).
- Autosegmental graphs with Inertia (and NCC) are totally ordered sets of *embracements* (Corollary 1).
- An embracement is a connected component with the ≤-embraced isolates.
- There are three infinite codes (Thm. 4-6) for the graphs generated by the embracements.
- The embracements are coded (and decoded) to and from the visual code in deterministic linear time (Thm. 7).
- There are finite state bijections between the codes (Thm. 8-9; Cor. 2).
- These bijections are not arbitrary, but preserve the structure of the graphs (Thm. 10).
- The iconicity of the prior bracket-based code Yli-Jyrä (2013; 2015) is expected to apply also to its current symmetric extension.
- Many practically important extensions to the codes are feasible although not yet elaborated.

Acknowledgements

The work has been done under the grant #270354 of the Academy of Finland and partly during my Clare Hall Visiting Fellowship 2013-2014 allowing me to visit the Cambridge University. I would like to express special thanks to Francis Nolan, Arvi Hurskainen, Andy Black, Lotta Aunio, Miikka Silfverberg, and Steven Bird for inspiring discussions.

References

Daniel M. Albro. 1994. *Amar: a computational model of Autosegmental Phonology*. Ph.D. thesis, Massachusetts Institute of Technology.

Bruce Bagemihl. 1989. The crossing constraint and 'backwards languages'. *Natural Language & Linguistic Theory*, 7(4):pp. 481–549.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications.

Jean Berstel, Dominique Perrin, and Christophe Reutenauer. 2010. *Codes and Automata*, volume 129 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press.

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. 1990. Phonological events. *Journal of Linguistics*, 26:33–56.

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

H. Andrew Black. 1997. TonePars: A computational tool for exloring autosegmental tonology. SIL Electronic Working Papers 1997-007, December 1997. Summer Institute of Linguistics.

Julie Carson-Berndsen. 1998. *Time Map Phonology: Finite State Models and Event Logics in Speech Recognition*. Kluwer Academic Publishers, Dordrecht.

Farida Cassimjee and Charles W. Kisseberth. 1998. Optimal domains theory and Bantu tonology: A case study from Isixhosa and Shingazidja. In Larry Hyman and Charles W. Kisseberth, editors, *Theoretical Aspects of Bantu Tone*. CSLI, Stanford.

George. N. Clements. 1985. The geometry of phonological features. *Phonology Yearbook*, 2:225–252.

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

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

Charles L. Hamblin. 1962. Translation to and from Polish notation. *Computer Journal*, 5:210–213.

Susan Hertz. 1990. The Delta programming language: an integrated approach to non-linear phonology, phonetics and speech synthesis. In John Kingston and Mary E. Beckman, editors, *Papers in Laboratory Phonology 1*. Cambridge University Press.

Arvi Hurskainen. 2009. Enriching text with tone marks: An application to Kinyarwanda language. Technical Reports in Language Technology 4, University of Helsinki. `http://www.njas.helsinki.fi/salama`.

Adam Jardine. 2013. Logic and the generative power of autosegmental phonology. In *Proceedings of Phonology 2013*.

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

D. Robert Ladd. 2008. *Intonational Phonology*. Cambridge Studies in Linguistics. Cambridge University Press.

William Leben. 1973. *Suprasegmental phonology*. Ph.D. thesis, MIT. Published by Garland Publishing, New York, 1980.

John J. McCarthy and Alan Prince. 1996. *Prosodic Morphology 1986*. Number 13 in Linguistics Department Faculty Publication Series. University of Massachusetts - Amherst, MA. `http://scholarworks.umass.edu/linguist_faculty_pubs/13`.

John J. McCarthy. 2011. Autosegmental spreading in Optimality Theory. Paper 27, University of Massachusetts – Amherst, Amherst.

Jackson Muhirwe. 2010. Morphological analysis of tone marked Kinyarwanda text. In *Finite-State Methods and Natural Language Processing*, volume 6062 of *Lecture Notes in Computer Science*, pages 48–55. Springer Berlin Heidelberg.

Derek Nurse and Gérard Philippson. 2003. *The Bantu Languages*. Routledge, New York.

David Odden. 1986. On the role of the Obligatory Contour Principle in phonological theory. *Language*, 62:353–383.

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

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press.

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

Moira Yip. 2002. *Tone*. Cambridge Studies in Linguistics. Cambridge University Press.

Anssi Yli-Jyrä. 2013. On finite-state tonology with autosegmental representations. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, pages 90–98, St. Andrews, UK. Association for Computational Linguistics.

Anssi Yli-Jyrä. 2015. Building lexical transducers from autosegmental grammars: a practical methodology. Manuscript. 57 pages. Submitted to Journal of Language Modelling.