Three Equivalent Codes for Autosegmental Representations

Anssi Yli-Jyrä
Department of Modern Languages, University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

A string encoding for a subclass of bipartite graphs enables graph rewriting used in autosegmental descriptions of tone phonology via existing and highly optimized finite-state transducer toolkits (Yli-Jyrä 2013). The current work offers a rigorous treatment of this code-theoretic approach, generalizing the methodology to all bipartite graphs having no crossing edges and unordered nodes. We present three bijectively related codes each of which exhibit unique characteristics while preserving the freedom to violate or express the OCP constraint. The codes are infinite, finite-state representable and optimal (efficiently computable, invertible, locally iconic, compositional) in the sense of Kornai (1995). They extend the encoding approach with visualization, generality and flexibility and they make encoded graphs a strong candidate when the formal semantics of autosegmental phonology or non-crossing alignment relations are implemented within the confines of regular grammar.

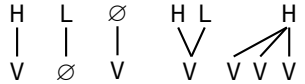
1 Introduction

1.1 Autosegmental Phonology

Historically, Autosegmental Phonology (Goldsmith, 1976) helped to articulate the *independence of the melody*, the *decomposition of contour tones as level tones* and the *Obligatory Contour Principle (OCP)* that bans adjacent copies of phonological units. Its multi-tiered design was also a crucial step towards more complex phonological theories such as Feature Geometry (Clements, 1985), Optimal Domains Theory (Cassimjee and Kisseberth, 1998) and Autosegmental-Metrical Theory (Ladd, 2008). Moreover, autosegmental representations are timelessly interesting formal objects as they are similar to certain alignment relations arising in statistical machine translation.

Autosegmental Phonology continues to play an appreciated role in language documentation efforts for under-resourced languages. In particular, describing Bantu tone is extremely complex (Nurse and Philippson, 2003) as the lexical, morphological, phonological and syntactic dimensions of each language are intertwined in unique ways although there are tonological characteristics shared across the Bantu family. Many prior accounts have relied on autosegmental rules while constraint-based approaches sometimes offer an equivalent solution only too painfully via varying constraint rankings. After all, languages with a young or developing orthography urgently need *lexical transducers* (Beesley and Karttunen, 2003) and, especially, *tonologically enriched lexical transducers* (Hurskainen, 2009; Muhirwe, 2010) that can be used to improve speech applications and various text normalization tools. It is, therefore, desirable that tone processing is incorporated into existing and highly optimized finite-state toolkits and lexical databases that have been primarily designed for segmental phonology and morphology.

According to Autosegmental Phonology, phonological 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 clarity, we naively discard the actual phonological and metrical values of tone bearing units *(TBUs)* in the segmental tier. A possible way to associate the tone string `HLHLH` with the segmental string `VVVVV` is given by the autosegmental representation



(1)

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

1. **"mobility":** The alignment between tones and segments is mutable.
2. **"stability of tone":** A tone can be temporarily unattached to a segment.
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.

Until recently, autosegmental processing appeared to require advanced computational representations and a graph rewriting system. Consequently, the lack of a usable finite-state implementations has remained as a bottleneck.

1.2 Prior Computational Approaches

A few computational implementation approaches for autosegmental representation have been proposed. For example, there are software systems such as the Delta programming language (Hertz, 1990), AMAR (Albro, 1994) and Tone Pars (Black, 1997). These systems do not explicitly rely on automata, but there are other computational approaches that are clearly related to finite automata and transducers:

1. OCP-bound finite-state approaches (Bird and Klein, 1990; Bird and Ellison, 1994; Bird, 1995; Carson-Berndsen, 1998; Jardine, 2013) hardwire the Obligatory Contour Principle into their mathematical formalization of autosegments. The principle was originally motivated by Leben (1973), but its status as a universal constraint has been under scrutiny ever since (see McCarthy 1986).

2. OCP-neutral finite-state approaches (Kornai, 1995; Wiebe, 1992; Yli-Jyrä, 2013) see that the OCP is sometimes violated unless not maintained by the rules (Odden, 1986). OCP-neutral approaches have been advocated (Goldsmith, 1976; Odden, 1986; Prince and Smolensky, 2004) and it is sometimes assumed by field linguistic descriptions (Halme, 2004).

Both approaches have their merits. Since our interest is in extending the capacity of generic finite-state toolkits towards tonology rather than to develop a principled phonological theory, the OCP-neutral approach is now preferred.

The main prior contributions towards the implementation of the OCP-neutral approach can be summarized as follows:

- Wiebe (1992) and Kornai (1995) model autosegmental representations via multi-tape automata and the corresponding codes.

- Yli-Jyrä (2013) adopts some simplifying assumptions and then models the representations as bracketed strings.

The bracket encoding of Yli-Jyrä (2013) is practically implemented and compatible with the standard technology for building lexical transducers. Unfortunately, it supports only a particular subset of all two-tiered autosegmental representations.

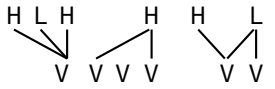
1.3 The Limits of the Prior Work

In the bracket-based encoding of 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 $\overline{\text{H}}\text{V}\text{H}\,\overline{\text{L}}\text{L}\,\text{V}\,\overline{\text{H}}\text{V}\text{L}\,\overline{\text{H}}\text{VVV}\text{H}$.

It is not difficult to see that the encoding scheme has some limitations that prompt further research:

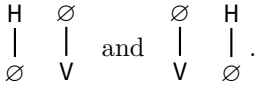
- The approach is not expressed and analyzed in terms of coding theory. Therefore, the issue of existence of a *compositional coding function* (Kornai, 1995) was not properly addressed.

- 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 such as:

In the encoding, there is also a limitation that is mainly useful. Namely, the encoding does not equate the autosegmental representations



Instead, the order of unlinked elements is meaningful: $\overline{\text{H}}\text{H}\text{V} \neq \text{V}\overline{\text{H}}\text{H}$. This property that we now call INERTIA means that the order of isolated vertices is fixed by the way in which they are concatenated and changed only by force: via rewriting. As the shuffling of floating tones and toneless segments is now delegated to the rules, the OCP-neutral representations are fully linearizable although the representations discussed by Wiebe (1992) and Kornai (1995) are not.

1.4 The Results of This Paper

This paper sticks to two-tiered autosegmental representations since they can be used to build more complex multi-tier representations. However, our approach is more closely tied to the coding theory (Berstel et al., 2010) in order to get formal results that are fundamental to a practical finite-state implementation such as Yli-Jyrä (2013).

The first consequence (Section 3) of this approach is that there is an **optimal coding function** (Kornai, 1995) for inertial autosegmental graphs that are generated by an appropriately defined *infinite source "alphabet"*.

The second major result is that a *serial code with an optimal coding function can also be visually attractive* (Section 4). Our **visual code** is easy to read and write. This is handy as no separate visualization step is required to make the code human readable.

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).

We will also see that the visual code and the symmetric bracket code are bijectively related via a **Polish code** that is a simple, computationally motivated code (Section 6).

2 Definitions

2.1 Autosegmental Graphs

Different axiomatizations for autosegmental graphs have been proposed since Goldsmith (1976). In particular, the NON-CROSSING CONSTRAINT (NCC) is a common assumption in autosegmental theory, although it has been contested in prosodic morphology (Bagemihl, 1989; McCarthy and Prince, 1996).

For us, the tone-segment associations are formalized as bipartite graphs whose vertices are (auto)segments. The sets of vertices are ordered in each graph since they correspond to tonal and segmental strings. In addition, the associating edges between them are ordered due to the NCC. The vertices without incident edges are called *isolates* and they are ordered due to the INERTIA property that was implicit already in Yli-Jyrä (2013) but is now formalized for the first time.

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) has formalized computational implementation of autosegmental graphs as a 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, the computation is carried out in real time, e.g. 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: $\hat{\beta}(GG') = \hat{\beta}(G)\hat{\beta}(G')$.[1]

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 $i \in \{1, .., n\}$.

Observe that if the concatenation does not add any new edges to disjoint graphs under the operation, there must be a separate code element for every connected graph. Since the set of connected graphs is infinite, a block code is not possible, but the code must be *infinite*. In addition, possible isolates intervening any pair of nodes of a connected component must be coded along with the component.

An infinite code is perfectly normal notion (Berstel et al., 2010) although it requires more than a

[1] Beware the terminology: a coding function β can be *compositional* (Kornai, 1995) without being *composable* (Berstel et al., 2010).

finite table (for example, a graph serialization algorithm). For example, the *maximal semaphore code* $X = A^*S - A^*SA^+$ for a *semaphore set* $S \subseteq A^+$ is infinite at least when $S \subset A$. Furthermore, any infinite or finite subset of some infinite code is also a code and can be sufficient when the source text is limited — e.g., a finite data set or a specific natural language.

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, and
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 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 connected components 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 $\leq_S$-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 distinct 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 vertices s and t are connected and there are no crossing edges, they are connected to vertices u and u'. This means, however, that $C = C'$; 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.

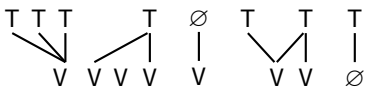
4 New Codes for the Graphs

4.1 The Visual Code

The visual code is defined over the alphabet

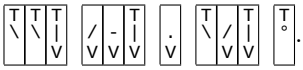
$$\begin{aligned} A =& \{(\texttt{/V}), (\texttt{\ T}), (\texttt{|VT}), (\texttt{-V}), (\texttt{.V}), \\ & (\texttt{- T}), (\texttt{° T})\} \\ =& \left\{ \begin{bmatrix} \texttt{/} \\ \texttt{V} \end{bmatrix}, \begin{bmatrix} \texttt{T} \\ \texttt{\textbackslash} \end{bmatrix}, \begin{bmatrix} \texttt{T} \\ \texttt{|} \\ \texttt{V} \end{bmatrix}, \begin{bmatrix} \texttt{-} \\ \texttt{V} \end{bmatrix}, \begin{bmatrix} \texttt{.} \\ \texttt{V} \end{bmatrix}, \begin{bmatrix} \texttt{T} \\ \texttt{-} \end{bmatrix}, \begin{bmatrix} \texttt{T} \\ \texttt{°} \end{bmatrix} \right\}. \end{aligned}$$

The idea is that autosegmental graphs such as



(2)

are encoded as visually similar strings:



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[2]

$$X = \left\{ \begin{smallmatrix}\mathsf{.}\\ \mathsf{V}\end{smallmatrix} \bigcup \begin{smallmatrix}\mathsf{T}\\ \mathsf{°}\end{smallmatrix} \right\} \bigcup \left\{ \begin{smallmatrix}\mathsf{/}\\ \mathsf{V}\end{smallmatrix} \begin{smallmatrix}\mathsf{-}\\ \mathsf{V}\end{smallmatrix}^{*} \bigcup \begin{smallmatrix}\mathsf{T}\\ \backslash\end{smallmatrix} \begin{smallmatrix}\mathsf{T}\\ \mathsf{-}\end{smallmatrix}^{*} \right\}^{*} \begin{smallmatrix}\mathsf{T}\\ \mathsf{|}\\ \mathsf{V}\end{smallmatrix}.$$

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

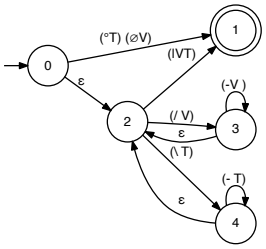


Figure 1: Recognizer for the visual code

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 $B = \{\mathsf{[}, \mathsf{]}, \overline{\mathsf{V}}, \mathsf{V}, \mathsf{°}\}$. 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

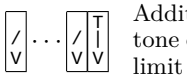
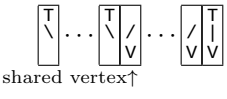
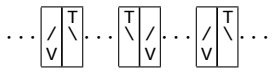
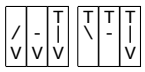
$$\overline{\mathsf{V}}\mathsf{[][][V]}\ \ \overline{\mathsf{V}}\mathsf{[V}\overline{\mathsf{V}}\mathsf{°V}\overline{\mathsf{V}}\mathsf{V]}\ \ \overline{\mathsf{V}}\mathsf{V}\ \ \overline{\mathsf{V}}\mathsf{[][V}\overline{\mathsf{V}}\mathsf{V]}\ \ \mathsf{[]}.$$

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

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

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

Table 1: Encoding a graph with the visual code

	Non-embraced isolated vertices are indicated by respective symbols that are ordered according to the order of the isolates.
	The rightmost edge of each non-trivial tree is indicated by a bar.
	Additional vowels can be linked to the tone of the rightmost vertice (the natural limit is, however, three tones).
	Additional tones can be linked to the vowel of the rightmost vertice.
 shared vertex↑	Any shared vertex starts a new subtree.
	There is no limit for the depth of chained subtrees.
	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). □

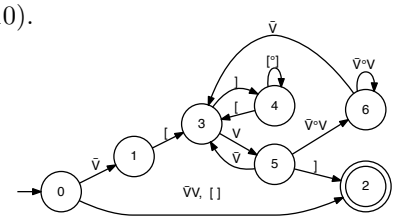


Figure 2: Recognizer for the bracket code

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 $D = \{\mathsf{|}, \mathsf{°}, \mathsf{V}, \mathsf{T}\}$. For example, (2) is encoded as

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

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

$[\,] = \boxed{\begin{smallmatrix}\mathsf{T}\\ \circ\end{smallmatrix}}$, $\overline{\mathsf{V}}\mathsf{V} = \boxed{\begin{smallmatrix}\cdot\\ \mathsf{V}\end{smallmatrix}}$	Non-embraced isolates correspond to simple brackets.
$\overline{\mathsf{V}}[\mathsf{V}] = \boxed{\begin{smallmatrix}\mathsf{T}\\ \mid\\ \mathsf{V}\end{smallmatrix}}$	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.
$\overbrace{\overline{\mathsf{V}}[\mathsf{V}\overline{\mathsf{V}}\mathsf{V}\cdots\ \overline{\mathsf{V}}]}^{\text{gr.dtr's tree}}\overbrace{[\,]\cdots[\mathsf{V}\overline{\mathsf{V}}\cdots\mathsf{V}]}^{\text{topmost tree}}$ (daughter's 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.

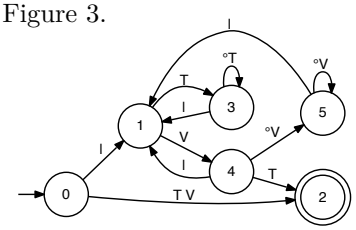


Figure 3: Recognizer for the Polish code

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

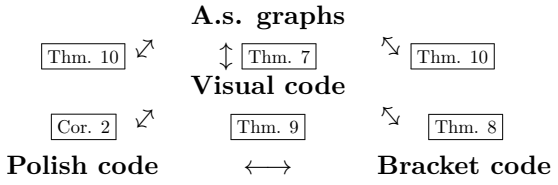


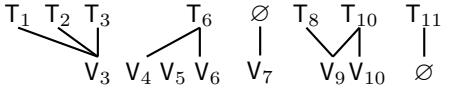
Figure 4: The equivalence results

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 directed graph contains cycles they are between $=_F$ -equivalent vertices. In order to be able to view the graph as an acyclic graph and sort it topologically, we temporarily eliminate each cycle by unifying the vertices in it. This results into a graph with position indexes:

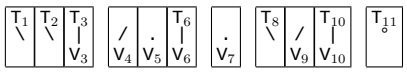


For example, T_3 and V_3 in this graph are are viewed as a single vertex and assigned to a single position during the sorting procedure. Using the position indexes of the vertices, we encode each position with an element of the alphabet A of the visual code. In total, the encoding is computed in deterministic 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 num-

bers 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.

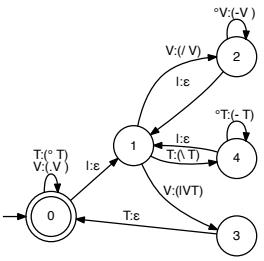


Figure 5: $Y^* \leftrightarrow X^*$

The transducer in Figure 5 is constructed with the 2-tape regular expression[3]:

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

□

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.

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

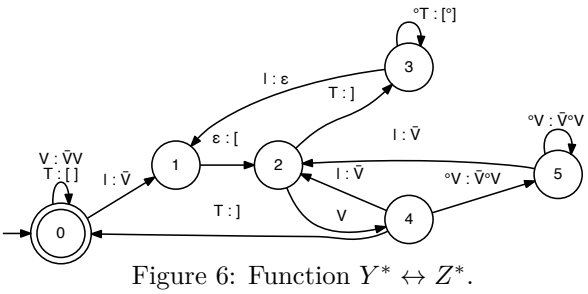


Figure 6: Function $Y^* \leftrightarrow Z^*$.

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

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

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:

$$\begin{array}{c} \mathsf{|T°T|VT\ \ |V°V|VT\ \ V\ \ |T|V|VT\ \ T} \Rightarrow \\ \mathsf{|[][°]|[V]\ \ |[V°V|V]\ \ V\ \ |[]|[V|V]\ \ []} \Rightarrow \\ \mathsf{\bar{V}[][°]|[V]\ \ \bar{V}[V\bar{V}°V\bar{V}V]\ \ \bar{V}V\ \ \bar{V}[]|[V\bar{V}V]\ \ []} \Rightarrow \\ \mathsf{\bar{V}[][°][V]\ \ \bar{V}[V\bar{V}°V\bar{V}V]\ \ \bar{V}V\ \ \bar{V}[][V\bar{V}V]\ \ []}. \end{array}$$

□

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 The Results

The current work links the coding theory to linearizations for autosegmental graphs and obtains the following results:

- An autosegmental graph with INERTIA (and NCC) consists of segments called *embracements* (Corollary 1) each of which consists of a connected component and the corresponding $\leq$-embraced isolates.
- There is a *compositional coding function* for the infinitely generated set of inertial autosegmental graphs (Thm. 1). In particular, the coding function for the visual code is *computable* and *invertible* in deterministic linear time (Thm. 7).
- In fact, there are multiple infinite codes (Thm. 4-6) that are related via finite state bijections (Thm. 8-9; Cor. 2) with the visual code. The bijections are not arbitrary, but preserve the structure of the coded graphs (Thm. 10). In fact, all the codes presented in this paper are, in a certain sense, *iconic*.
- The symmetric bracket code is a generalization of a bracket code that has been presented informally and applied already successfully to finite-state compilation of tonological grammar components (Yli-Jyrä, 2013).

6.2 Further Work

The current presentation has not discussed finer distinctions between the vertexes or how to use the codes to build actual applications. Fortunately, some extensions and applications are almost immediate by analogy with the prior similar work (Yli-Jyrä, 2013).

- **Encoded Graph Processing.** The current work contributes to processing encoded graphs via string transducers. Previously, Yli-Jyrä (2013) has demonstrated an approach that manipulates regular languages consisting of encoded autosegmental graphs using a readily available finite-state transducer toolkit. This application of coding theory seems to contain many fresh problems. Since the encoded graphs are strings, their languages can be closed under such operations as union, concatenation, quotients, Kleene star, and difference. It would be interesting to see what else one can do with the regular languages of encoded graphs.
- **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 have different sets of brackets for different level tones.
- **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

- **Adding More Tiers.** In order to capture Feature Geometry and more elaborated multi-tier representations, we should extend the current codes into n-partite graphs. This extension seems feasible under a tree-shaped feature geometry.

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 Arvi Hurskainen,

Andy Black, Lotta Aunio, Francis Nolan, Miikka Silfverberg, and Steven Bird for inspiring discussions and the reviewers for many useful comments.

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 L. Hyman and C. 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 J. Kingston and M. 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.

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.

John J. McCarthy. 1986. OCP effects: Gemination and antigemination. *Linguistic Inquiry*, 17:207–263.

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 FSMNLP*, pages 90–98, St. Andrews, UK. Association for Computational Linguistics.