Phonology **32** (2015). Supplementary materials

Long words in maximum entropy phonotactic grammars

Robert Daland
University of California, Los Angeles

Supplementary materials

These supplementary materials introduce three theorems which formalise the central contributions of this paper. The general purpose of the theorems is discussed in the paper for the lay linguist. This introduction briefly describes the theorems for more mathematically inclined readers.

In linguistic terms, Theorem 1 demonstrates that every phonotactic grammar can be represented by a score function mapping strings to the non-positive integers. Here, to be 'represented' has the technical meaning that if the grammar distinguishes string x as strictly less well-formed than x, then x is mapped to a smaller (more negative) integer (whereas if they are not different in well-formedness, they are mapped to the same integer). The proof critically relies on the property that there cannot be an infinite sequence of strings in which well-formedness increases. The proof uses the notions of equivalence relation and their corresponding equivalence classes, which may be unfamiliar to many linguists; otherwise, it largely consists of showing that various ordering properties are preserved by relevant mappings.

Theorem 2 gives SUFFICIENT conditions for a maxent phonotactic grammar to have a finite partition function. This in turn means that it assigns a well-defined probability distribution over all possible strings. (If the partition function is not finite, string probabilities are undefined.) The proof draws on the theory of geometric series, the laws of exponentiation/logarithms and basic facts about cardinality/counting, but should be accessible to most mathematically inclined linguists.

While Theorem 2 gives sufficient conditions for a finite partition function, Theorem 3 gives NECESSARY conditions. The starting point is previous research from formal language theory demonstrating how constraint-based mappings that instantiate regular relations can be represented by (weighted) finite-state automata. This means that maxent HG grammars can be given a convenient graphical representation. Next, the proof draws on the notion of EXPECTATION SEMI-RING, formalised in Eisner (2002), which associates a probability and a violation vector with each node. Finally, the paper draws on the tight coupling between weighted graphs and matrices to derive a PENALTY MATRIX P, where the entry P_{ij} represents the change in well-formedness incurred by transitioning from node j to node i of the finite-state automaton, expressed as a multiplicative factor. If the principal eigenvalue of this matrix has a magnitude greater than 1, then the partition function is not finite.

1 Theorem 1

Definition: A relation $\precsim$ on a set X is a PREWELLORDERING if and only if

$\precsim$ is TOTAL
- $\forall x, y \in X$, $x \precsim y$ or $y \precsim x$
- 'every pair of strings is comparable'

$\precsim$ is TRANSITIVE
- $\forall x, z \in X$, $x \precsim y$ and $y \precsim z$ implies $x \precsim z$
- e.g. if [lb] is worse then [bn], and [bn] is worse than [bl], then [lb] is worse than [bl]

X is WELLORDERED by $\precsim$
- $\forall S \subset X$, $S \neq \emptyset \rightarrow \exists s_{max} \in S \ni \forall s \in S$, $s \precsim s_{max}$
- 'every non-empty subset of X has a maximal element'
- in other words, there cannot be an infinite sequence of strings that increase in well-formedness (though an infinite decreasing sequence is allowed)

(Note that this inverts the normal direction for wellordering; it is more typical to require a minimal element, but the formulations are completely analogous.)

Definition: A SCORE FUNCTION f for a set X is a map from elements of X to a totally ordered space $(K, <_K)$. A score function f represents the prewell-ordering $\precsim$ on X if

$\forall x, y \in X$, $x \precsim y$ and $y \precsim x$ implies $f(x) = f(y)$
- 'equally well-formed strings are assigned equal scores'

$\forall x, y \in X$, $x \precnsim$ y implies $f(x) <_K f(y)$
- if string x is strictly less well-formed than y, x is assigned a strictly lower score than y

Theorem 1: Let Σ be an alphabet, and Σ^* the set of finite strings over Σ. Every prewellordering $\preceq_{\Sigma^*}$ over Σ^* can be represented by a score function $f : \Sigma^* \to (\mathbb{N}^-, <)$ where $\mathbb{N}^-$ is the non-positive integers and $<$ is the normal order on integers.

Proof: Given the prewellordering $\preceq_{\Sigma^*}$, define the relation $\sim$ by $x \sim y$ if and only if $x \preceq_{\Sigma^*} y$ and $y \preceq_{\Sigma^*} x$ (the 'equality relation'). Symmetry follows from the definition of $\sim$, while transitivity and reflexivity follow from the transitivity and reflexivity of prewellorders; thus $\sim$ is an equivalence relation. Let K denote the set of equivalence classes of $\sim$, and $f_\sim$ the map from elements of X (strings) to their equivalence class, $f_\sim(x) = [x]$. Because Σ^* is countable, K must be countable. *Claim:* $\preceq_{\Sigma^*}$ induces a total strict well-ordering on K. Define $[x] <_K [y]$ if and only if $\exists w \in [x]$, $z \in [y]$ such that $w \preceq_{\Sigma^*} z$ but $[x] \neq [y]$. Totality, transitivity, strictness and antisymmetry follow, so $<_K$ is a strict total order. Wellordering follows from the well-ordering of $\preceq_{\Sigma^*}$, so $(K, <_K)$ is totally strictly wellordered.

Now the task is to construct a map from $(K, <_K)$ to the non-positive integers. Proceed by induction. *Base case:* By wellordering, there must exist a value $k_0 \in K$ that is maximal according to $<_K$. Then define $\phi(k_0) = 0$. *Induction step:* Now suppose there is a sequence of values $\{k_i\}_{i=1}^n$ such that $\phi(k_i) = -i \; \forall_i$, $(0 \leq i \leq n)$. Let $K_{n+1} = K \setminus \{k_i\}_{i=1}^n$. Since K_{n+1} is a subset of K, by wellordering it has a maximal element k_{n+1}. Then define $\phi(k_{n+1}) = -(n+1)$. This completes the induction step.

Since K is countable, ϕ must be well-defined for every $k \in K$, and $\phi : K \to \mathbb{N}^-$. Evidently, $k_x <_K k_y$ implies $\phi(k_x) < \phi(k_y) \; \forall k_x, k_y \in K$ by construction. It is also easy to show that $\phi(k_x) < \phi(k_y)$ implies $k_x <_K k_y$ for all $k_x, k_y \in K$. Observe that $\phi(k_x) < \phi(k_y)$ implies that K_x is a proper subset of K_y. Since the maximal element k_y of K_y was removed to construct K_x, it follows that $k <_K k_y \; \forall_k \in K_x$. Since $k_x \in K_x$, it follows that $k_x <_K k_y$. In short, ϕ preserves all and only the order differences of $<_K$.

Two maps have been defined: $f_\sim : \Sigma^* \to K$ maps every string to a well-formedness equivalence class, and $\phi : K \to \mathbb{N}^-$ maps each well-formedness equivalence class to a non-positive integer score. Then their composition $f(x) = \phi(f_\sim(x))$ maps every string to a non-positive score, $f_\sim : \Sigma^* \to \mathbb{N}^-$ with the following properties:

$$\forall x, y \in \Sigma^*, \; x \preceq_{\Sigma^*} y \text{ and } y \preceq_{\Sigma^*} x \Leftrightarrow f(x) = f(y)$$

$$\forall x, y \in \Sigma^*, \; x \precnsim_{\Sigma^*} y \Leftrightarrow f(x) < f(y)$$

In other words, f represents the prewellordering $\preceq_{\Sigma^*}$ using the non-negative integers.

2 Theorem 2

Definition: A HARMONIC GRAMMAR is a tuple $G = (X, Y, R, \vec{C}, \vec{w})$ where

- X is a set of LEXICAL REPRESENTATIONS
- Y is a set of SURFACE REPRESENTATIONS
- $R \subset X \times Y$ indicates the set of possible surface representations that a given lexical representation can map to, called the CANDIDATES: $candidates(x) = \{y \in Y \mid xRy\}$
- $\vec{C}$ is a finite set of CONSTRAINT FUNCTIONS $\vec{C} = \{C_k\}_{k=1}^{K}$, where each constraint function assigns a non-negative integer violation count to lexical-surface pairs, $\forall_k$, $C_k : R \to \mathbb{N}$
- $\vec{w}$ is a set of real-valued, non-positive weights $\vec{w} = \{w_k\}_{k=1}^{K}$, associated with the constraints

The HARMONY of a lexical–surface pair $H_{\vec{C},\vec{w}}(x, y)$ is defined as the weighted sum of the constraint violations, $H(x, y) = \sum_{k=1}^{K} w_k \cdot C_k(x, y)$. Given some $x \in X$, surface representation $y \in candidates(x)$ is a WINNER if and only if $\nexists z \in candidates(x)$, such that $H(x, z) > H(x, y)$, in other words if y has the greatest harmony value among the candidates of x. We write $G(x) = y$ in case y is a unique winner for x. Thus G denotes a mapping from X to Y just in case every $x \in X$ has a unique winner $G(x) \in Y$.

Definition: A MAXENT HG grammar is a harmonic grammar $G = (X, Y, R, \vec{C}, \vec{w})$ that is augmented with a WELL-FORMEDNESS FUNCTION $\Phi : R \to \mathbb{R}^+$ defined by $\Phi(x, y) = e^{H(x,y)}$ (the range of the exponentiation is the non-negative reals $\mathbb{R}^+$). The PARTITION FUNCTION associated with a lexical representation x, $Z : X \to \mathbb{R}^+ \cup \{\infty\}$ is defined as the sum of the well-formedness values of all of x's candidates, $Z(x) = \sum_{xRy} \Phi(x, y)$. In case the partition function is finite for a given x, G defines the conditional probability distribution $Pr(y \mid x) = \Phi(x, y)/Z(x)$.

Definition: A MAXENT PHONOTACTIC GRAMMAR over Σ^* is a maxent HG grammar $G = (X, Y, R, \vec{C}, \vec{w})$ in which X is a singleton set $\{\omega\}$, $Y = \Sigma^*$ where Σ^* is the set of finite strings over some alphabet Σ, and $\omega R y \; \forall y \in \Sigma^*$. Since X and R are trivial, it is convenient to omit them from the notation: $G = (\Sigma^*, \vec{C}, \vec{w})$, $H(y) = \sum_{k=1}^{K} w_k \cdot C_k(y)$, etc.

Definition: Define the *STRUCT constraint *STRUCT : $\Sigma^* \to \mathbb{N}$ by *STRUCT$(\sigma) = k$ if and only if $\sigma \in \Sigma^k$. In other words, the number of *STRUCT violations that a string σ incurs is simply the length of σ.

Theorem 2: Let $G = (\Sigma^*, \vec{C}, \vec{w})$ be a maxent phonotactic grammar. If the constraint set $\vec{C}$ includes a *Struct constraint (i.e. $\exists i < k$ such that C_i = *Struct) and $w_{*\text{Struct}} < -ln|\Sigma|$, then the partition function Z of G is finite.

Proof: The proof goes in two steps. The first step is to consider a maxent phonotactic grammar G' over Σ^* whose constraint set includes *only* *Struct, and prove that $w_{*\text{Struct}} < -ln|\Sigma|$ implies that Z' is finite. The second step is to show that $\Phi'(\sigma)$ is an upper bound for $\Phi(\sigma)$ for all $\sigma \in \Sigma^*$, so Z' is an upper bound for Z. Finitude of Z follows from finitude of Z'.

Part 1: Given $G = (\Sigma^*, \vec{C}, \vec{w})$, define the maxent phonotactic grammar G' over Σ^* to have the sole constraint $\vec{C} = \{*\text{Struct}\}$ with weight w; the corresponding harmony and well-formedness functions are denoted H' and Φ'. We will abuse notation by defining $\Phi'(X) = \sum_{\sigma \in X} \Phi'(\sigma)$, i.e. extending the definition of the well-formedness function so that it measures sets of strings (rather than merely evaluating individual strings). The goal is to compute Z' by partitioning Σ^* by the length of its strings. Let us begin by considering the value $\Phi'(\Sigma^k)$ for an arbitrary k. By definition, $\sigma \in \Sigma^k$ implies *Struct(σ) = k, so that $H'(\sigma) = w \cdot *\text{Struct}(\sigma) = w \cdot k$ and $\Phi'(\sigma) = e^{H'(\sigma)} = e^{w \cdot k} = (e^w)^k$. The number of strings in Σ^k is $|\Sigma|^k$, and each such string σ has a constant well-formedness value, $\Phi'(\sigma) = (e^w)^k$. Then the total $\Phi'(\Sigma^k)$ can be expressed as the product, $\Phi'(\Sigma^k) = |\Sigma^k| \cdot (e^w)^k = (|\Sigma| \cdot e^w)^k$. Now it is possible to express Z' as the sum over the partition by length ($\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$):

$$\begin{aligned}\Phi'(\Sigma^*) &= \sum_{k=0}^{\infty} \Phi'(\Sigma^k) \\ &= \sum_{k=0}^{\infty} (|\Sigma| \cdot e^w)^k\end{aligned}$$

This latter is instantly recognisable as an infinite series with common ratio $r = |\Sigma| \cdot e^w$. The theory of infinite series indicates that this value is finite (and well-defined) if and only if $|r| < 1$. Since $|\Sigma|$ and e^w are positive, this reduces to the inequality $|\Sigma| \cdot e^w < 1$, or $e^w < |\Sigma|^{-1}$. The natural logarithm is an increasing function, which can be applied to both sides to preserve the inequality, yielding $w < -ln|\Sigma|$.

Part 2: We begin by showing that $\forall \sigma \in \Sigma^*$, $H(\sigma) \leq H'(\sigma)$. Without loss of generality, reassign indices so that $H(\sigma) = w \cdot *\text{Struct}(\sigma) + \sum_{k=1}^{K-1} \cdot C_k(\sigma)$ and recall that $H'(\sigma) = w \cdot *\text{Struct}(\sigma)$. Then $H(\sigma) - H'(\sigma) = \sum_{k=1}^{K-1} w_k \cdot C_k(\sigma)$ for all σ. Now since $w_k \leq 0 \; \forall k$ and $C_k \geq 0 \; \forall k$, it follows that $w_k \cdot C_k(\sigma) \leq 0$ $\forall k$, and therefore the sum must be non-positive as well, i.e. $H(\sigma) - H'(\sigma)$ $\leq 0 \; \forall \sigma \in \Sigma^*$. But this means that $H'(\sigma)$ is an upper bound for $H(\sigma)$ for every σ. And since exponentiation is an increasing function, this implies that $\Phi'(\sigma)$ is an upper bound for $\Phi(\sigma)$ for all σ. And since Z' is the sum of Φ' values, while Z is the sum of Φ values, this means that Z' is an upper bound for Z. Thus, if Z' is finite, Z must be as well. This completes the proof.

3 Theorem 3

Riggle (2009) gives a technical but well-exemplified presentation of finite-state Optimality Theory. The essence of this approach is to model constraints as (possibly weighted) finite-state transducers. In order to understand Theorem 3, it is necessary to understand how maxent phonotactic grammars are implemented as finite-state transducers, and in order to understand this, it is first necessary to understand a little about finite-state transducers.

Intuitively, a finite-state transducer (FST) is a computational device which accepts one 'input' string and converts it to another 'output' string. Finite-state machines are conventionally represented as directed graphs. The nodes of the graph represent 'states' of the machine. Arcs represent the transition from one state to another; normally, this is associated with reading one symbol from the input string, and writing one symbol to the output string. The particular mapping that an arc represents is indicated with a label on the arc; the label *x*:*y* indicates that the arc transduces input symbol *x* into output symbol *y*. The 'start' and 'final' nodes of the graph are graphically indicated by one or more conventions. Here I use the convention that the start state is indicated by an arrow from nowhere pointing to the start node, while the end state is indicated by a double circle. Figure 1 gives a finite-state transducer that implements an intervocalic voicing rule. The FST in Figure 1 only accepts the input sequence /apa/, and if it is presented with that input sequence, it yields the output sequence [aba]. Traversing a single arc in the graph represents reading/writing one symbol, so that the nodes/states of the graph represent the amount of the input string that has been read. It is possible for there to be multiple arcs emanating from or leading to a state. For example, the FST in Fig. 1 could be augmented with an additional arc 'b:b' pointing from node 2 to node 3; the modified FST would then implement two mappings, /apa/ → [aba] and /aba/ → [aba].

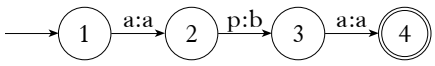

Figure 1
FST implementing the mapping /apa/ → [aba].

Riggle (2009) uses simple weighted finite-state transducers to represent individual constraints. A grammar can then be represented using an operation known as 'intersection'. Figure 2, which is slightly adapted from Riggle (2009: Fig. 1), illustrates three faithfulness constraints and their intersection. Each transition *x*:*y* is followed by a value indicating the amount or type of violation that is incurred. For example, when an input symbol maps to itself – symbolised by *X*:*X* - no faithfulness violation is incurred, and this is symbolised by $\overline{0}$. The symbol *ε* is a special symbol indicating the empty string. Thus when an empty input is mapped to a *V*, the grammar should record a DEP(V) violation in some way. Similarly, when a non-empty input symbol is mapped to an empty output, the input symbol has been deleted, and the grammar should record a MAX violation. The intersection of the three simple FSTs in Fig. 2 is represented in the rightmost FST. In that FST, any ALTERATION to the input is penalised except for transducing an empty input into a syllable boundary (the **.** symbol).

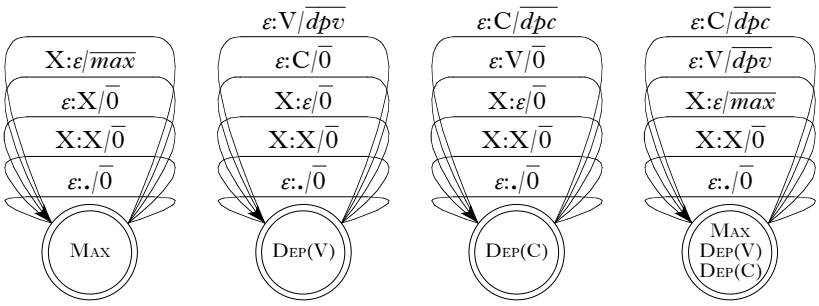

Figure 2
Simple weighted FSTs representing MAX, DEP(V) and DEP(C), and their intersection

Riggle further illustrates by generating the FST for a full CV grammar, containing the hard constraint ((*C*)*V*(*C*).)*, as well as the violable constraints ONSET (*ons*) and NOCODA (*noc*), and the three faithfulness constraints from before. The full CV grammar, slightly adapted from Riggle (2009: Fig. 3), is reproduced in Fig. 3. The reader is encouraged to consult Riggle (2009) for additional exposition.

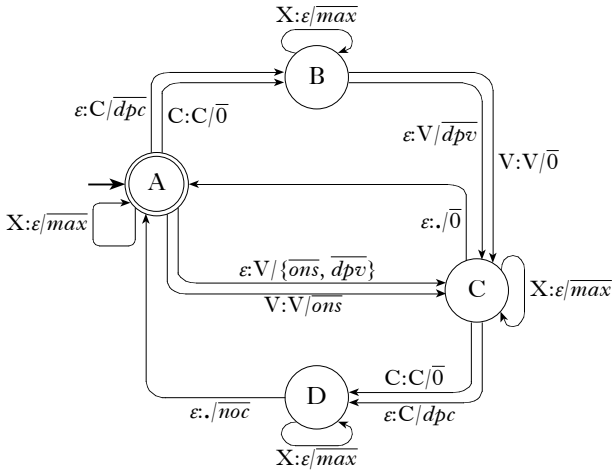

Figure 3

Syllable grammar. State A represents the start of a syllable. State C represents the moment after the nucleus has been written to the output. State B represents the moment after an onset consonant has been written to the output, whether it was in the input or epenthesised. State D represents the moment after a postvocalic consonant has been read from the input, whether it was written to the output or deleted. Only state A is a final state, and this is why the links from C and D to A contain the end-of-syllable symbol **.**. Thus, an onsetless syllable [V**.**] is generated by the path $A \rightarrow C \rightarrow A$, while a [CV**.**] syllable is generated by the path $A \rightarrow B \rightarrow C \rightarrow A$.

The final background material is Eisner's (2002) notion of an EXPECTATION SEMI-RING.

In natural-language processing applications – where FSTs are most widely used – it is typical to annotate arcs with WEIGHTS, expressed in either the probability or log-probability domain. For example, the CV grammar in Figure 3 could be expressed as a probabilistic FST by assigning the symbol $\overline{0}$ a pseudo-probability of 1, while $\overline{max}$, $\overline{dpv}$, $\overline{dpc}$, $\overline{ons}$ and $\overline{noc}$ would be assigned pseudo-probabilities less than 1. Pseudo-probabilities would then be normalised by requiring that the probability of leaving a non-final state must sum to 1. The probability of a path is defined as the product of the probabilities of traversing all the arcs in the path, and the probability of an input–output pair is the sum over the probabilities of all distinct paths that generate the pair.

In fact, Eisner points out that most useful formulations for weighted FSTs have a common, abstract character: the weights of paths obey the axioms of a CLOSED SEMI-RING. A closed semi-ring $(K, \oplus, \otimes, *)$ is a mathematical structure which is a slight generalisation of the natural numbers: it consists of a set K with an 'addition' operation $\oplus$ (associative, commutative, identity element $\overline{0}$), a 'multiplication' operation (associative, distributes over $\oplus$ from the right and the left, identity element $\overline{1}$), and 'closure' for infinite loops ($k^* \equiv_{def} \bigoplus_{i=1}^{\infty} k^i$). Eisner (2002) states: 'ordinary probabilities fall in the semi-ring $(\mathbb{R}_{\geq 0}, +, \times, *)$'.

In Riggle's (2009) formulation, the symbols $\overline{0}$, $\overline{max}$, $\overline{dpv}$, $\overline{dpc}$, $\overline{ons}$ and $\overline{noc}$ are treated as unit VECTORS, so that the violation vector for a path may be accumulated as the sum the unit violation vectors encountered along the path. (Technically, Riggle construes violations as belonging to a multiset with a semi-ring structure, but for the present purposes these are the equivalent.) Given that multiple paths may generate the same input–output pair, but with possibly different violation vectors, one should also define how to calculate the violation vector for an input–output pair. From a total ranking over constraints, Riggle defines this as the violation profile of the path which has the minimum number of violations over the highest-ranked constraint on which there is a difference. Riggle refers to this formulation as a 'violation semi-ring', and points to its similarities with the TROPICAL SEMI-RING over the real numbers.

Eisner's expectation semi-ring combines both probabilities and violation vectors. In Eisner's own words:

> Abstractly, let us say that each path π has not only a probability $P(\pi) \in [0, 1]$ but also a VALUE $val(\pi)$ in a vector space V, which counts the arcs, features or coin flips encountered along path π. The value of a path is the sum of the values assigned to its arcs... The idea is to augment the weight data structure with expectation information, so each weight records a probability *and* a vector counting the parameters that contributed to that probability.

Eisner proceeds to define the V-EXPECTATION SEMI-RING $(\mathbb{R}_{\geq 0} \times V, \oplus, \otimes, *)$ as follows:

$$(p_1, v_1) \otimes (p_2, v_2) \equiv_{def} (p_1 p_2, p_1 v_2 + v_1 p_2)$$
$$(p_1, v_1) \otimes (p_2, v_2) \equiv_{def} (p_1 + p_2, v_1 + v_2)$$
$$\text{if p* defined, } (p, v)^* \equiv_{def} (p^*, p^* v p^*)$$

He elaborates:

> If an arc has probability p and value p, we give it the weight (p, pv) so that our invariant (see above) holds if [the pathset] Π consists of a single length-0 or length-1 path. The above definitions are designed to preserve our invariant as we build up larger paths and pathsets. $\otimes$ lets us concatenate (e.g.) simple paths π_1, π_2 to get a longer path π with $P(\pi) = (\pi_1)P(\pi_2)$ and $val(\pi) = val(\pi_1) + val(\pi_2)$. The definition of $\otimes$ guarantees [the invariant] that path π's weight will be $(P(\pi), P(\pi) \cdot val(\pi))$. $\oplus$ lets us take the union of two disjoint pathsets, and * computes infinite unions.

Now that the expectation semi-ring has been explained, we are in position to consider its application to maxent HG grammars. As stated in Hayes & Wilson (2008: 389):

> the properties of a very large set of strings can be computed by representing the set as a finite state machine. We construct our machines by first representing each constraint as a weighted finite state acceptor. Using intersection …, we then combine the constraints into a single machine that embodies the full grammar … Each path through this machine corresponds to a phonological representation together with its vector of constraint violations. We then obtain the $E[C_i]$ values by summing over all paths through the machine …

Theorem 3 is concerned with whether the FST associated with a maxent phonotactic grammar has well-defined expectations, or in other words, whether it has a finite partition function. It does this by explicitly constructing a matrix which sums over the WELL-FORMEDNESS VALUES of every path through the FST. In order to understand this, it is first necessary to understand how matrix multiplication can represent accumulation over paths through a matrix, and how the principal eigenvalue reflects important aspects of the limit behaviour.

Technically, a directed graph G consists of some nodes (indexed 1 to n), and directed edges which point from a node j to some node i. The ADJACENCY MATRIX (A_{ij}) associated with G is an $n \times n$ matrix in which:

$$A_{ij} = \begin{cases} 1 & \text{there is an edge from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

Clearly $(A^1)_{ij}$ indicates the number of paths of length 1 from node j to node i. It turns out that something much more general is true: $(A^m)_{ij}$ indicates the number of paths of length m from node j to node i. To see this, consider the (i, j) entry of A^{m+1}. Since $A^{m+1} = A^m \cdot A$, $(A^{m+1})_{ij}$ must be the dot product of the ith row of A^m and the jth column of A: $(A^{m+1})_{ij} = \sum_{k=1}^{n} (A^m)_{ik} \cdot A_{kj}$. The value $(A^m)_{ik}$ indicates the number of paths of length exactly m from k to i, and A_{kj} indicates the number of paths of length exactly 1 from j to k. Thus, $(A^m)_{ik} \cdot A_{kj}$ indicates the number of paths of length $m + 1$ that begin at j and end at i, passing through k as the first step. Since every path from j to i must pass through some first step, the total number of paths of length $m + 1$ is given by their sum across k. Therefore, $(A^{m+1})_{ij}$ indicates the total number of paths of length $m + 1$ from node j to node i. In this case, matrix multiplication represents the 'accumulation' of the number of paths. The principal eigenvalue represents the (limit of the) RATIO by which the number of paths grows as the path length is increased. In fully connected graphs, this value will be n (the number of nodes); in acyclic graphs (where it is not possible to return to any state i once it has been visited), the principal eigenvalue of the adjacency matrix is 0.

Another example in which the principal eigenvalue reflects 'accumulation' or 'flow' of some value across a graph is in Markov chains. A Markov chain consists of an initial distribution over states and a TRANSITION MATRIX (T_{ij}), in which the value T_{ij} indicates the probability of visiting state i next, given that the current state is j. Since probabilities must be non-negative and sum to 1, this enforces the condition that the sum over each column of T must be 1. Very similarly to the adjacency matrix, $(T^m)_{ij}$ indicates the probability of ending in state i after m steps, given that one started in state j. The principal eigenvalue of the Markov chain always has the value 1, indicating that there is a constant 'amount' of probability which simply flows across the graph. Markov chains are very well-studied; for a more thorough treatment the reader is directed to Grimstead & Snell (1997: ch. 11).

In the present case, we wish to determine whether a given maxent HG grammar has a finite partition function. The first step is to translate the grammar into a weighted FST with a violation semi-ring, as in the work of Riggle and Eisner. The next step it to use matrix multiplication to characterise the 'accumulation' of well-formedness values of paths through the FST, using a PENALTY MATRIX. It follows that the partition function is finite if and only if the principal eigenvalue of this matrix has a magnitude of 1 or less.

Definition: Let $G = (\Sigma^*, \vec{C}, \vec{w})$ be a maxent phonotactic grammar, and $G' = (\Sigma, Q, q_0, F, E, \rho)$ be the associated weighted FST over V, where

- Σ is the output alphabet
- Q is the set of states
- $q_1 \in Q$ is the unique start state
- $F \subset Q$ is the set of final states
- $E \subset (Q \times \Sigma \times V \times Q)$ is a finite set of transitions, each signifying the traversal from a state q_j to a state q_i after encountering an output symbol σ, and having a violation vector $\vec{c}_{ij}$ indicating the constraints that are violated in this transition

Let the states in G' be indexed $\{q_i\}_{i=1}^{n}$, and the constraints in G be indexed $\{C_k\}_{k=1}^{K}$. Then define an $(n+1) \times (n+1)$ PENALTY MATRIX P as follows:

$$P_{ij} = \begin{cases} \Phi(\vec{w} \cdot \vec{c}_{ij}) & \text{there is an edge from } q_j \text{ to } q_i \\ 1 & i = n+1 \text{ and } q_j \in F, \text{ i.e. } j \text{ is final} \\ 1 & i = n+1 \text{ and } j = n+1 \\ 0 & \text{otherwise} \end{cases}$$

The second and third rows of this definition are equivalent to adding a final final state q_{n+1} and requiring all final states to have a cost-free transition to the final final state, so that all well-formedness will accumulate in q_{n+1}.

Theorem 3: Let (P_{ij}) be the penalty matrix for a maxent phonotactic grammar $G = (\Sigma^*, \vec{C}, \vec{v})$ and its associated FST $G' = (\Sigma, Q, q_0, F, E, \rho)$. G has a finite partition function if and only if the principal eigenvalue of P has a magnitude of 1 or less.

Proof: The proof goes in two steps. The first is to show that every accepted string corresponds to a path beginning at q_1 and being 'absorbed' in q_{n+1}. The second is to show that $(P^m)_{ij}$ indicates the well-formedness value of all paths of length exactly m that begin at state j and end at state i (so that the well-formedness of all paths will 'accumulate' in q_{n+1}). It follows from this that $Z = \lim_{m\to\infty}(P^m)_{(n+1)(n+1)}$, which is finite if and only if the principal eigenvalue has magnitude 1 or less.

Step 1: The definition of FST gives that every path begins in the start state q_1, and ends in one of a set of final states $q_j \in F$. Then, the requirement that $P_{(n+1)j} = \delta_{ij}$ amounts to adding a cost-free transition to the extra state $n+1$, but only from pre-existing finals. The requirement that $P_{(n+1)(n+1)} = 1$ means that every string path which enters this state stays there without penalty. (The cost-free self-link is necessary to preserve the well-formedness so that it can be counted in the limit, rather than allowing it to vanish after the path is accepted.) Since every string that is accepted by the FST must begin at the start state and end at a final state, every terminating path of length $m-1$ through the FST is counted by the matrix entry $(P^m)_{(n+1)1}$ (and all subsequent values of m).

Step 2: Suppose that the violation vector $\vec{e}_{ik}$ is incurred by traversing a single path from q_k to q_i, and $\vec{e}_{kj}$ is incurred by traversing a single path from q_j to q_k. Then the well-formedness value associated with the $k \to i$ arc is $\Phi(\vec{v} \cdot \vec{e}_{ik}) = e^{\vec{v} \cdot \vec{e}_{ik}}$, and the well-formedness value associated with the j to k arc is $\Phi(\vec{v} \cdot \vec{e}_{kj}) = e^{\vec{v} \cdot \vec{e}_{kj}}$. Their product is $\Phi(\vec{v} \cdot \vec{e}_{ik}) \cdot \Phi(\vec{v} \cdot \vec{e}_{kj}) = e^{\vec{v} \cdot \vec{e}_{ik}} \cdot e^{\vec{v} \cdot \vec{e}_{kj}} = e^{\vec{v} \cdot \vec{e}_{ik} + \vec{v} \cdot \vec{e}_{kj}} = e^{\vec{v} \cdot (\vec{e}_{ik} + \vec{e}_{kj})} = \Phi(\vec{v} \cdot (\vec{e}_{ik} + \vec{e}_{kj}))$. In other words, the penalty is the same whether the violations are added up and assessed all at once, or assessed one by one. This is the same kind of 'invariant' as Eisner (2002) refers to. Now, it is already true by construction that $(P^1)_{ij}$ represents the penalty to a string incurred by transitioning from state j to state i in exactly 1 step. Now suppose that $(P^m)_{ij}$ represents the penalty incurred by transitioning from state j to state i in exactly m steps. Then $(P^{m+1})_{ij} = \sum_{k=1}^{n}(P^m)_{ik} \cdot (P_{kj})$. Then the well-formedness of all paths of exactly length $m+1$ is decomposed just as with the adjacency matrix counting paths above.

It follows from these that the value $(P^m)_{(n+1)1}$ represents the sum of the well-formedness values of all accepted strings of length $m-1$ or less. Since the partition function is defined as the sum of the well-formedness values of all terminating strings, it follows that $Z = \lim_{m\to\infty}(P^m + 1)_{(n+1)1}$. Since all of the well-formedness must eventually accumulate in this cell for finite strings, it follows that Z is finite if and only if the principal eigenvalue of P has magnitude less than 1.