THE FRANK–SATTA THEOREM

Definition 1. A weighted language over $\Sigma \times W$ is a function $c : \Sigma^* \rightarrow W$, where $W$ is the range of possible weights.

In what follows, we are going to use $\mathbb{N}$ for $W$. Next week, we will state exactly what kinds of sets we want to allow for $W$, and we will see that, if we pick the right sort of $W$, we can state all the weighted constraints that we’re interested in by just adding weights (elements of $W$) to the arcs of a finite-state automaton, to obtain a weighted finite-state automaton. We will also be able to generalize this to weighted relations, for which we will use weighted finite-state transducers.

For today, the important bit of leverage is going to come from assuming that, for any given weight $v$, the set $\{ w \in \Sigma^* | c(w) = v \}$ is a regular language. Before we make use of this though, we need to see where we are going to make use of it.

Definition 2. The set of optimal strings in $L$ under a weighted language $c$, $\text{Optim}_c(L)$, is the set $\{ w \in L | c(w) = \min_{w \in L} c(w) \}$.

Here’s what an OT grammar looks like:

Definition 3. An optimality system (OS) is a triple $G = \langle \Sigma, \text{Gen}, \text{Con} \rangle$, where $\Sigma$ is a finite alphabet, $\text{Gen}$ is a relation over $\Sigma^* \times \Sigma^*$ and $\text{Con} = \langle c_1, \ldots, c_p \rangle$, $p \geq 1$, is an ordered sequence of weighted language over $\Sigma \times \mathbb{N}$.

And here’s how Eval works:

Definition 4. Any optimality system $G$ with $p$ elements in $\text{Con}$ induces, for each $0 \leq i \leq p$, a map $\text{Eval}_G^i : \Sigma^* \rightarrow 2^{\Sigma^*}$, which gives the set of candidates that remain after filtering the potential candidates up to constraint $i$, as follows:

$$\text{Eval}_G^i(w) = \begin{cases} \text{Gen}(w) & i = 0 \\ \text{Eval}_G^{i-1}(w) & \text{if } i \geq 1 \text{ and } \text{Optim}_{c_i}(\text{Eval}_G^{i-1}(w)) = \text{Eval}_G^{i-1}(w) \\ \text{Optim}_{c_i}(\text{Eval}_G^{i-1}(w)) & \text{if } i \geq 1 \text{ and } \text{Optim}_{c_i}(\text{Eval}_G^{i-1}(w)) \neq \text{Eval}_G^{i-1}(w) \end{cases}$$

We call $\text{Eval}_G^p$ simply $\text{Eval}_G$, to indicate that it is the maximal mapping induced by $G$.

The idea is that (apart from the $\text{Gen}$ step), we proceed at each step by either filtering the output of the last step to be optimal, or passing everything from the last step, if we do not rule out anything new (second case). This conflates two cases which for us are intuitively different: the case where the constraint does not apply to any of the candidates, and so they are all given a pass by default; and the case where there is only one candidate remaining, i.e. $\text{Eval}_G^{i-1}(w)$ has only one element, and so whatever constraint $i$ does, it always assigns to that one element the trivial minimum weight.

Theorem 5. (Frank–Satta Theorem). Suppose an OS $G = \langle \Sigma, \text{Gen}, \text{Con} \rangle$ is such that $\text{Gen}$ is a regular relation, each $c \in \text{Con}$ is finite in the sense that its codomain, $\{ v | \exists w \in \Sigma^*. c(w) = v \}$, is finite, and, for each $c \in \text{Con}$, the restriction of $\Sigma^*$ to those strings mapping to any particular $v$ in its codomain is regular. Then $\text{Eval}_G$ is a regular relation.
Proof. We start with a simpler case: suppose that each \( c \in Con \) is finite with codomain of size exactly 2. Without loss of generality we can conflate the codomain of every \( c \) to \( \{\bot, \top\} \), where \( \bot \) stands in for the maximum (NB!) weight assigned by \( c \) to any \( w \), and \( \top \) for the minimum. We denote by \( L_c \) the set of all \( w \) such that \( c(w) = \top \), and note that \( L_c \) is a regular language. In this case, \( \text{Eval}^1_G \) reduces to the following three cases:

\[
\text{Eval}^1_G(w) = \begin{cases} 
\text{Gen}(w) & i = 0 \\
\text{Eval}^1_G(w) & \text{if } i \geq 1 \text{ and } \text{Eval}^{i-1}_G(w) \cap L_{c_i} = \emptyset \\
\text{Eval}^{i-1}_G(w) \cap L_{c_i}(w) & \text{if } i \geq 1 \text{ and } \text{Eval}^{i-1}_G(w) \cap L_{c_i} \neq \emptyset
\end{cases}
\]

These three cases do not align with the three cases given in the definition of \( \text{Eval} \): \( \text{Optim}_{c_i}(\text{Eval}^{i-1}_G(w)) = \text{Eval}^{i-1}_G(w) \) does not imply that \( \text{Eval}^1_G(w) \cap L_{c_i} = \emptyset \). On the one hand, it could be that all candidates in \( \text{Eval}^1_G(w) \) receive \( \bot \) when constraint \( c_i \) is applied, in which case it is indeed true that \( \text{Eval}^1_G(w) \cap L_{c_i} = \emptyset \); however, it could also be the case that all candidates in \( \text{Eval}^1_G(w) \) receive \( \top \), in which case \( \text{Eval}^1_G(w) \cap L_{c_i} = \text{Eval}^{i-1}_G(w) \), and in general this is not \( \emptyset \). But in this case we nevertheless also pass only \( \text{Eval}^{i-1}_G(w) \). (On the other hand, \( \text{Eval}^1_G(w) \cap L_{c_i} = \emptyset \) does imply that \( \text{Optim}_{c_i}(\text{Eval}^{i-1}_G(w)) = \text{Eval}^1_G(w) \), because, in order for the latter to be true it must be the case that for all elements in \( \text{Eval}^1_G(w) \) are assigned the same weight by \( c_i \), while in order for the former to be true it must be the case that for all elements in \( \text{Eval}^1_G(w) \) are assigned the weight \( \bot \) by \( c_i \)—and thus all the same weight.)

If instead of \( L_{c_i} \) we consider the corresponding constraint (in the relation sense) \( \text{Id}(L_{c_i}) \), then this is equivalent to the following:

\[
\text{Eval}^1_G(w) = \begin{cases} 
\text{Gen}(w) & i = 0 \\
\text{Eval}^{i-1}_G(w) & \text{if } i \geq 1 \text{ and } \text{Eval}^{i-1}_G \circ \text{Id}(L_{c_i})[w] = \emptyset \\
\text{Eval}^{i-1}_G \circ \text{Id}(L_{c_i})[w] & \text{if } i \geq 1 \text{ and } \text{Eval}^{i-1}_G \circ \text{Id}(L_{c_i})[w] \neq \emptyset
\end{cases}
\]

We can make quick work of the question of whether \( \text{Eval}^1_G \circ \text{Id}(L_{c_i})[w] = \emptyset \iff \text{Eval}^1_G \cap L_{c_i} = \emptyset \): if \( \text{Eval}^{i-1}_G \circ \text{Id}(L_{c_i})[w] = \emptyset \), then \( \exists x. \text{Eval}^{i-1}_G(w) = x \& x \in L_{c_i} \), and thus \( \text{Eval}^1_G \cap L_{c_i} = \emptyset \), and conversely.

We now proceed, as usual, by proving the stronger statement which holds for all \( i \), so that we can proceed by induction; that is, we show by induction on \( i \) that, for all \( 0 \leq i \leq p \), \( \text{Eval}^i_G \) is regular. If \( i = 0 \) the claim is true by hypothesis, since for \( i = 0 \) \( \text{Eval}^0_G \) is simply \( \text{Gen} \). Now suppose that \( \text{Eval}^k_G \) is a regular relation. Clearly in the case where \( \text{Eval}^k_G \circ \text{Id}(L_{c_k+1})[w] = \emptyset \), \( \text{Eval}^{k+1}_G = \text{Eval}^k_G \) is regular. And, since, by hypothesis, \( L_{c_i} \) is a regular language, \( \text{Eval}^k_G \circ \text{Id}(L_{c_{k+1}})[w] \) is also regular by closure under composition. Thus the restriction of \( \text{Eval}^{k+1}_G \) to either one of these two subsets of \( \Sigma^* \) is regular, and therefore the union of these two restricted relations is regular.

In order to extend this to the general case of finite weighted languages in \( Con \), we apply the following construction to obtain an equivalent set of binary weighted languages: without loss of generality we can assume that every \( c \in Con \) has codomain \( \{0, \ldots, k\} \), for some \( k > 1 \). Now for any \( c, j, 1 \leq j \leq k \), let \( \langle c, j \rangle(w) \) be a new weighted language such that \( \langle c, j \rangle(w) = \top \) if \( c(w) < j \), \( \langle c, j \rangle(w) = \bot \) if \( c(w) \geq j \). Since \( L_{c, j} = \bigcup_{m=0}^{j-1} L_m \), by hypothesis and by closure under finite union we can deduce that \( L_{c, j} \) is regular and thus its complement is regular and thus \( \langle c, j \rangle \) will always have the property that the restriction of \( \Sigma^* \) to those strings mapping to any particular \( v \) in its codomain is regular.

Some important notes: countable union of regular languages is not regular, and is in general recursively enumerable (consider taking the union of all the singleton sets containing exactly one element of any given language \( L \)). Also, remember that constraints need access to both the input and the output, and therefore \( \text{Eval} \) is not a mapping from underlying representations to surface representations. It is a mapping from underlying representations to underlying–surface pairs. However, it is easy enough to come up with an encoding of such
a pair as a string such that the input could be stripped away at the end by composition with a finite-state transducer: for example, interleave input and output symbols in $Gen$, and then keep only every other symbol in the strings output by $Eval_G$. 