WEIGHTED FINITE-STATE TRANSDUCERS

Definition 1. A weighted relation is a function $R : \Sigma_i^* \times \Sigma_o^* \rightarrow W$, where $W$ is a set of possible weights.

As we said last week, we are going to use machines with weights added to the arcs to compute weighted languages and relations. We will also add weights to the initial state and to the final states. Now, we won’t always be using numbers as weights. But, in general, regardless of the number of tapes we have, we want to be able to compute the weight assigned to a particular input as follows:

- If there are no paths through the automaton labelled with the appropriate symbols, assign the input a special weight, $\overline{0}$
- If there is one path through the automaton labelled with the appropriate symbols, compute the weight by taking all the weights on all the arcs in that path and combining them in some way—some way that, like multiplication, will allow us to “annihilate” the whole path by having a single $\overline{0}$ anywhere along the path, and will allow us to include transitions along the path that don’t change the weight by giving them a special weight $1$; call the operation that does this combination $\otimes$
- If there are multiple paths through the automaton labelled with the appropriate symbols, compute the weight by doing the above for each path, and then combining them in some other way—some way that, like addition, will not annihilate the whole set of paths just because one of them has weight $\overline{0}$, but will rather take that element as an identity; call the operation that does this combination $\oplus$

For example, suppose we were to use $\mathbb{R}_{\geq 0}$ as the set of weights. Then we could use $0$ as $\overline{0}$, $1$ as $1$, $\times$ as $\otimes$, and $+$ as $\oplus$, and we would get:

$$w^p(\pi) = w_i \times \left[ \prod_{(q_f, i, w, q_i) \in \pi} w \right] \times w_f$$

where

- $w_f$ is the "final weight" associated with the last state in the path, $q_f$
- $w_i$ is the "initial weight" associated with the otherwise first state in the path, $q_i$

$$w^s(s) = \sum_{\pi \text{ accepting } s} w^p(\pi)$$

We can use the initial and final weights to make certain states initial states or final states. To take another example, however, suppose we use $\{\bot, \top\}$ as the set of weights. Then for $\otimes$ we could use $\land$, and for $\oplus$ we could use $\lor$. Then, interpreting $\bot$ as a way of “turning off” arcs and states—i.e. making arcs act as if they were not there, making certain states non-initial (by putting $\bot$ as an initial weight), and making certain states non-accepting (by putting $\bot$ as a final weight)—we could interpret $\bot$ as “reject” and $\top$ as “accept,” and we have defined precisely the same computations for computing these weights as we used in unweighted NFSAs.

In general, we want the weights to have the structure of a semiring.

Definition 2. A semiring $K$ is a 5-tuple $\langle K, \oplus, \otimes, 0, 1 \rangle$ such that:

1. $\oplus$ (“collection”) is associative (i.e. $a \oplus (b \oplus c) = (a \oplus b) \oplus c$)
2. $\oplus$ is commutative (i.e. $a \oplus b = b \oplus a$)

Date: May 7, 2012.
(3) \( \otimes \) (“extension”) is associative (i.e. \( a \otimes (b \otimes c) = (a \otimes b) \otimes c \))

(4) \( \overline{0} \) is an annihilator for \( \otimes \) (i.e. \( \overline{0} \otimes a = a \otimes \overline{0} = \overline{0} \))

(5) \( \overline{0} \) is an identity for \( \oplus \) (i.e. \( \overline{0} \oplus a = a \oplus \overline{0} = a \))

(6) \( \overline{1} \) is an identity for \( \otimes \) (i.e. \( \overline{1} \otimes a = a \otimes \overline{1} = a \))

(7) \( \otimes \) distributes over \( \oplus \) (i.e. \( a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c \) and \( (b \oplus c) \otimes a = b \otimes a \oplus c \otimes a \))

It’s easy enough to show that both \( \mathbb{R}_{\geq 0} \) and \( \{ \bot, \top \} \) can form semirings with the structure described above (the latter is called the “boolean semiring”).

We spell out the details of what we did before as follows.

**Definition 3.** A weighted finite state machine is a six-tuple \( \langle S, Q, K, \delta, \lambda, \rho \rangle \) (either transducer or acceptor depending whether \( S \) is symbols or pairs), where \( S \) is a finite alphabet, \( Q \) is a finite set of states, \( K \) is a semiring \( \langle \mathbb{K}, \oplus, \otimes, 0, 1 \rangle \), \( \delta : Q \times S \to \mathbb{K} \times Q \), \( \lambda : Q \to \mathbb{K} \) (the “initial weights”), and \( \rho : Q \to \mathbb{K} \) (the “final weights”).

Any WFSM will define a function from its input to the weights.

**Definition 4.** Given some WFSM \( A = \langle S, Q, K, \delta, \lambda, \rho \rangle \): a transition \( t \) is a four-tuple \( \langle q_1(t), s(t), k(t), q_2(t) \rangle \), where \( q_1(t) \) and \( q_2(t) \) are states, \( s(t) \) is an element of the alphabet, and \( k(t) \) is an element of \( K \). A path is a sequence of transitions \( t_1t_2\cdots t_N \) such that \( q_1(t_i) = q_2(t_{i-1}) \), for all \( 2 \leq i \leq N \). The label of a path \( s^p(\pi) = t_1t_2\cdots t_N \) is \( s(t_1)s(t_2)\cdots s(t_N) \). The weight of a path \( k^p(\pi) \) is \( k(t_1) \otimes k(t_2) \cdots \otimes k(t_N) \). A path is consistent with \( \delta \) iff for each transition \( t \) in \( \pi \), \( (k(t), q_2(t)) \in \delta(q_1(t), s(t)) \). Then \( A \) induces a mapping \( R : S^* \to \mathbb{K} \) such that \( R(s) = k \) iff there is a set of paths \( P \) consistent with \( \delta \), each with label \( s \), with \( \bigoplus_{\pi \in P} k^p(\pi) = k \). If \( S \) is a set of symbols, then \( R \) is a weighted regular language; if \( S \) is a set of pairs, then \( R \) is a weighted regular relation. The set of all sequences \( s \) in \( S^* \) such that \( R(s) \neq 0 \) is called the support of \( R \).

The support of any weighted regular language or relation is always regular: simply replace \( K \) with the boolean semiring and keep the \( \overline{0} \)s as is, replacing the other weights with \( \overline{1} \), and we obtain the classic unweighted accept/reject definitions.

Notice that in order to get the Frank and Satta result, we did almost exactly this, except slightly more general: we were given some weighted regular languages (each corresponding to an OT constraint), and we made them regular languages by repeatedly cutting the “number of violations” pie. We first cut the pie at \( 0 \) versus \( > 0 \), then at \( \leq 1 \) versus \( > 1 \), and so on. What we are doing to get the support is making just the first cut; but of course the point of the Frank–Satta idea is that we could make any replacement we like and the result would still be regular, and if we make this particular sequence of replacements, then we can reconstruct the original weighted language by simply reversing this construction and adding the weights we get back to an FSM for the language.

There is one idea you might be curious about:

**Definition 5.** Given two weighted relations \( R_1 : \Sigma_1 \times \Sigma_2 \to \mathbb{K} \) and \( R_2 : \Sigma_2 \times \Sigma_3 \to \mathbb{K} \), \( R_1 \circ R_2(x,y) \overset{\Delta}{=} \bigoplus_{s \in \Sigma_2} R_1(x,s) \otimes R_2(s,y) \).

Notice that the sum is over an infinite set; however, depending on the semiring, this sum may or may not be guaranteed to exist. For example, in the boolean semiring, it is guaranteed to exist. Over the natural numbers, however, it need not. Thus composition may or may not be available depending on the semiring.

However, this will turn out not to be the crucial thing. The crucial thing we will make use of is intersection.

**Definition 6.** Given two weighted relations \( R_1 : \Sigma_1 \times \Sigma_2 \to \mathbb{K} \) and \( R_2 : \Sigma_1 \times \Sigma_2 \to \mathbb{K} \), \( R_1 \cap R_2(x,y) \overset{\Delta}{=} R_1(x,y) \otimes R_2(x,y) \).

Now note that, just as for unweighted relations, intersection will only preserve regularity if the relations are same-length.