5. Weighted FSAs

1 Reminder/summary: (Boolean) FSAs

Generation of strings is fundamentally defined like this:

(1) $$x_1 x_2 \dots x_n \in \mathcal{L}(M) \iff \bigvee_{q_0 \in Q} \bigvee_{q_1 \in Q} \cdots \bigvee_{q_n \in Q} \Big[I(q_0) \wedge \Delta(q_0, x_1, q_1) \wedge \cdots \wedge \Delta(q_{n-1}, x_n, q_n) \wedge F(q_n) \Big]$$

Forward and backward are useful "helper functions" whose values can be computed recursively:

(2) $$\begin{aligned} \text{fwd}_M(\epsilon)(q) &= I(q) \\ \text{fwd}_M(x_1 \dots x_n)(q) &= \bigvee_{q_{n-1} \in Q} \Big[\text{fwd}_M(x_1 \dots x_{n-1})(q_{n-1}) \wedge \Delta(q_{n-1}, x_n, q) \Big] \end{aligned}$$

(3) $$\begin{aligned} \text{bwd}_M(\epsilon)(q) &= F(q) \\ \text{bwd}_M(x_1 \dots x_n)(q) &= \bigvee_{q_1 \in Q} \Big[\Delta(q, x_1, q_1) \wedge \text{bwd}_M(x_2 \dots x_n)(q_1) \Big] \end{aligned}$$

We can use forward and/or backward values to check whether a string is generated:

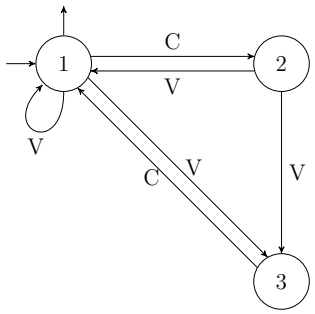
(4) $$uv \in \mathcal{L}(M) \iff \bigvee_{q \in Q} \Big[\text{fwd}_M(u)(q) \wedge \text{bwd}_M(v)(q) \Big]$$

(5) $$w \in \mathcal{L}(M) \iff \bigvee_{q \in Q} \Big[\text{fwd}_M(w)(q) \wedge \text{bwd}_M(\epsilon)(q) \Big] \iff \bigvee_{q \in Q} \Big[\text{fwd}_M(w)(q) \wedge F(q) \Big]$$

(6) $$w \in \mathcal{L}(M) \iff \bigvee_{q \in Q} \Big[\text{fwd}_M(\epsilon)(q) \wedge \text{bwd}_M(w)(q) \Big] \iff \bigvee_{q \in Q} \Big[I(q) \wedge \text{bwd}_M(w)(q) \Big]$$

Notice that everything here is about computing with *booleans*, using *conjunction* and *disjunction*.

(7)



What does this FSA say about the string 'CVC'?
What does this FSA say about the string 'VCV'?

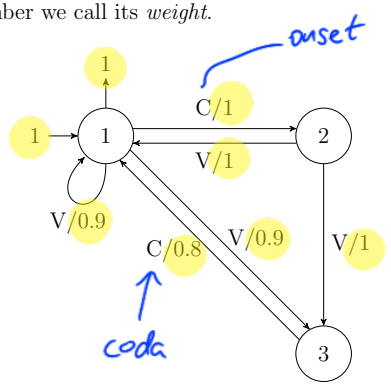
In both cases what it says is just "yes"/"true" — even though they are generated in different ways, 'VCV' is generated *two* ways and 'CVC' only one way, etc.

As soon as one choice of q_0, q_1, q_2, q_3 gives us a 'true', all the other choices of states make no difference.

2 Introducing weights

Here's a picture of a *weighted FSA*. Each possible "step", including starting and ending, has a non-negative number we call its *weight*.

(8)



Here's the linguistic idea being implemented:

- State 1 represents a syllable boundary.
- The 0.8 transition penalizes syllables that have a coda.
- The 0.9 transitions penalize syllables that don't have an onset.

We *multiply* the weights along a path. For example:

(9)
$$f_M(\text{CV}) = 1 \times 1 \times 1 \times 1$$
$$f_M(\text{VC}) = 1 \times 0.9 \times 0.8 \times 1$$
$$f_M(\text{CVC}) = 1 \times 1 \times 1 \times 0.8 \times 1$$
$$f_M(\text{CVV}) = 1 \times 1 \times 1 \times 0.9 \times 1$$
$$f_M(\text{VVC}) = 1 \times 0.9 \times 0.8 \times 0.9 \times 1$$

For "ambiguous" strings, we multiply along each path and take the *maximum*. The idea is that a string is as good as its best analysis.

(10)
for 'VCV' via path [1,1,2,1]: $1 \times 0.9 \times 1 \times 1 \times 1 = 0.9$
for 'VCV' via path [1,3,1,1]: $1 \times 0.9 \times 0.8 \times 0.9 \times 1 = 0.648$
$$f_M(\text{VCV}) = \max(0.9, 0.648) = 0.9$$

(11)
for 'CVCV' via path [1,2,1,2,1]: $1 \times 1 \times 1 \times 1 \times 1 \times 1 = 1$
for 'CVCV' via path [1,2,3,1,1]: $1 \times 1 \times 1 \times 0.8 \times 0.9 \times 1 = 0.72$
$$f_M(\text{CVCV}) = \max(1, 0.72) = 1$$

Any "impossible" transitions have a weight of zero. This makes things behave nicely:

- Any impossible paths get a weight of zero, because $0 \times x = 0$.
- Impossible paths "do nothing" to the overall value assigned to a string, because $\max(x, 0) = x$.

(12)
for 'VC' via path [1,3,1]: $1 \times 0.9 \times 0.8 \times 1 = 0.72$
for 'VC' via path [1,1,1]: $1 \times 1 \times 0 \times 1 = 0$
$$f_M(\text{VC}) = \max(0.72, 0) = 0.72$$

Question: What if every weight in a particular weighted FSA was either zero or one?

3 Weighted FSAs, formally

Now the careful definitions.

(13) A *weighted finite-state automaton* (WFSA) is a five-tuple $(Q, \Sigma, I, F, \Delta)$ where:

- Q is a finite set of states;
- Σ, the alphabet, is a finite set of symbols;
- $I : Q \to \mathbb{R}_{\geq 0}$ is a function assigning starting weights to states;
- $F : Q \to \mathbb{R}_{\geq 0}$ is a function assigning ending weights to states;
- $\Delta : (Q \times \Sigma \times Q) \to \mathbb{R}_{\geq 0}$ is a function assigning weights to transitions.

Then for any WFSA $M = (Q, \Sigma, I, F, \Delta)$, the function $f_M : \Sigma^* \to \mathbb{R}_{\geq 0}$ is defined as:

(14)
$$f_M(x_1 \dots x_n) = \max_{\text{all possible paths } p} \Big[\text{value for } x_1 \dots x_n \text{ via path } p\Big]$$
$$= \max_{q_0 \in Q} \max_{q_1 \in Q} \dots \max_{q_{n-1} \in Q} \max_{q_n \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n)\Big]$$

For the empty string, the length n is 0, so this just reduces to

(15)
$$f_M(\epsilon) = \max_{q_0 \in Q} [I(q_0) \times F(q_0)]$$

The definition in (14) makes the task of calculating $f_M(w)$ for long strings seem rather intimidating. It looks like you have to consider a very large number of distinct paths, and for each new path you consider you have to "start afresh", with no help from the values of other paths. Similarly, knowing the value of $f_M(u)$ and/or $f_M(v)$ doesn't seem to give you any head start in calculating the value of $f_M(uv)$.

4 Forward values

It turns out that the helpful idea of forward values carries over to weighted FSAs — although it is a bit less intuitive here.

Let's define $\text{fwd}_M(w)(q)$ as the best product-of-weights we can get by choosing some state to start at and then some transitions to take, which produce the string w and land us in the state q.

(16)
$$\text{fwd}_M(x_1 \dots x_n)(q) = \max_{q_0 \in Q} \max_{q_1 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q)\Big]$$

We can use forward values to calculate the overall weight $f_M(w)$ assigned to a string w:

(17)
$$f_M(w) = \max_{q_n \in Q} \Big[\text{fwd}_M(w)(q_n) \times F(q_n)\Big]$$

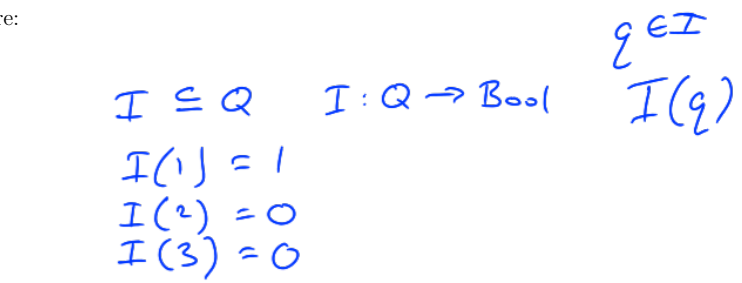
This equation is perhaps slightly less obvious than it looks. It relies on two important points:

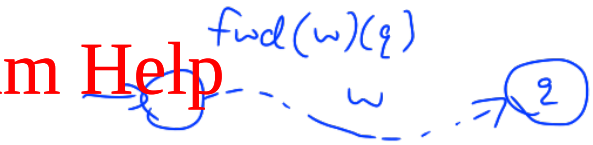
- The calculation of $\text{fwd}_M(w)(q_n)$ already took into account *all* the relevant ways of reaching state q_n, and *chose the very best of those*.
- A non-optimal way of getting-to-q_n can't be part of the optimal way of getting-to-q_n-and-stopping: if $a > b$, then $a \times F(q_n) > b \times F(q_n)$.

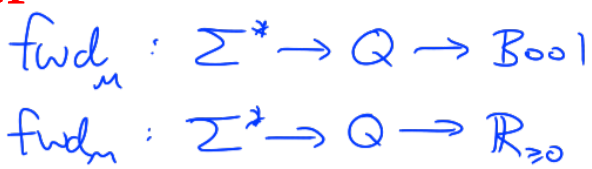
We can construct a table of forward values, just like we did for booleans.

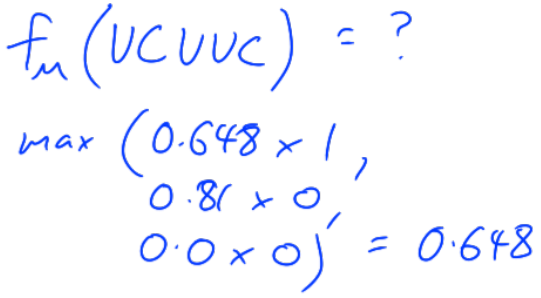
(18) Using the WFSA in (8):

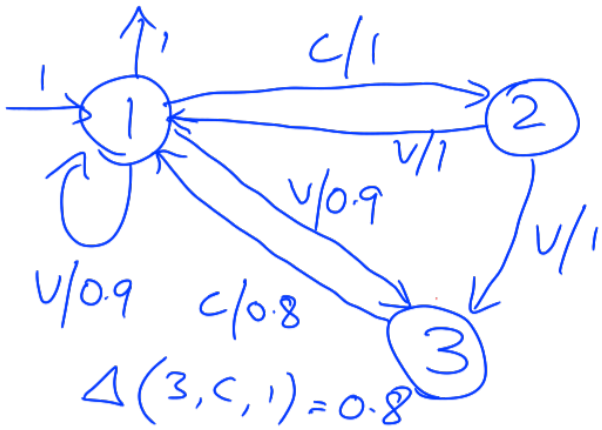
State	V	C	V	V	C	
1	1.0	0.9	0.72	0.9	0.81	0.648
2	0.0	0.0	0.9	0.0	0.0	0.81
3	0.0	0.9	0.0	0.9	0.81	0.0











As with booleans, the values in a column only depend on (i) the values in the column immediately to its left, and (ii) the last symbol being "added". So forward values can be computed recursively:

$$\text{fwd}_M(\epsilon)(q) = I(q) \tag{19}$$
$$\text{fwd}_M(x_1 \dots x_n)(q) = \max_{q_{n-1} \in Q} \Big[\text{fwd}_M(x_1 \dots x_{n-1})(q_{n-1}) \times \Delta(q_{n-1}, x_n, q) \Big]$$

It may take a bit of thought to convince yourself that this works, but the important logic is the same as for (17) above:

- The calculation of $\text{fwd}_M(x_1 \dots x_{n-1})(q_{n-1})$ already took into account *all* the relevant ways of reaching state q_{n-1}, and *chose the very best of those*.
- A non-optimal way of getting-to-q_{n-1} can't be part of the optimal way of getting-to-q_{n-1}-and-then-q: if $a > b$, then $a \times \Delta(q_{n-1}, x_n, q_n) > b \times \Delta(q_{n-1}, x_n, q_n)$.

More abstractly, the crucial underlying point here is that *multiplication distributes over max*, which says that we can "pull out" k in calculations like the following:

$$\max(a \times k, b \times k) = \max(a, b) \times k \qquad \max_{x \in S} \Big(g(x) \times k \Big) = \Big(\max_{x \in S} g(x) \Big) \times k \tag{20}$$

This is what justifies the following algebraic reshuffles, which get us from the definition in (14) to the equation in (17) (by "pulling out" $F(q_n)$):

$$\begin{aligned} f_M(x_1 \dots x_n) &= \max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \max_{q_n \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n) \Big] \\ &= \max_{q_n \in Q} \Big[\max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n) \Big] \Big] \\ &= \max_{q_n \in Q} \Big[\underbrace{\max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \Big]}_{\text{fwd}_M(x_1 \dots x_n)(q_n)} \times F(q_n) \Big] \end{aligned} \tag{21}$$

and from the definition in (16) to the equation in (19) (by "pulling out" $\Delta(q_{n-1}, x_n, q)$):

$$\begin{aligned} &\text{fwd}_M(x_1 \dots x_n)(q) \\ &= \max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q) \Big] \\ &= \max_{q_{n-1} \in Q} \Big[\max_{q_0 \in Q} \dots \max_{q_{n-2} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q) \Big] \Big] \\ &= \max_{q_{n-1} \in Q} \Big[\max_{q_0 \in Q} \dots \max_{q_{n-2} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-2}, x_{n-1}, q_{n-1}) \Big] \times \Delta(q_{n-1}, x_n, q) \Big] \\ &= \max_{q_{n-1} \in Q} \Big[\text{fwd}_M(x_1 \dots x_{n-1})(q_{n-1}) \times \Delta(q_{n-1}, x_n, q) \Big] \end{aligned} \tag{22}$$

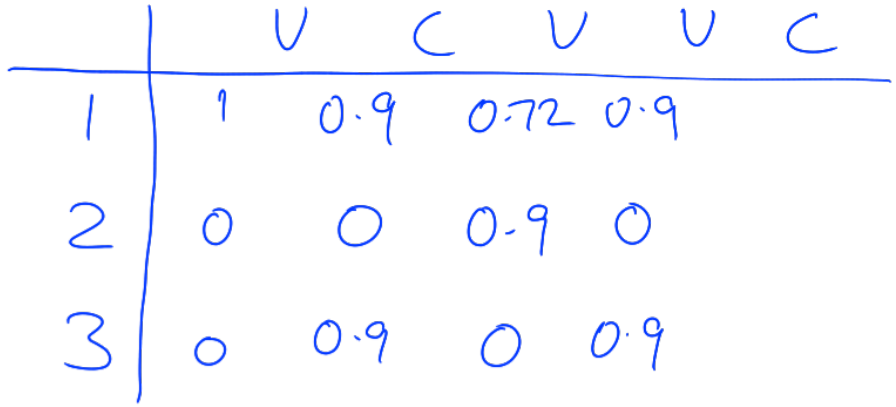
5 Probabilistic FSAs

If we are interested in defining a *probability distribution* over strings, then one simple idea is to take some WFSA M (say the one in (8) above), add up the weights of all generated strings and divide:

$$Z(M) = \sum_{w \in \Sigma^*} f_M(w) \qquad \Pr(w) = \frac{f_M(w)}{Z(M)} \tag{23}$$

But unfortunately this doesn't always work: sometimes the infinite sum $Z(M)$ does not converge.

A more common approach is to set up a WFSA so that it obeys certain conditions which guarantee that the sum $Z(M)$ turns out to be 1, so that $f_M(w)$ just is the probability of w; a WFSA that satisfies these conditions is called a *probabilistic finite-state automaton*.



$0.72 \times \Delta(1, V, 3) = 0.72 \times 0.9$

$0.9 \times \Delta(2, V, 3) = 0.9 \times 1$

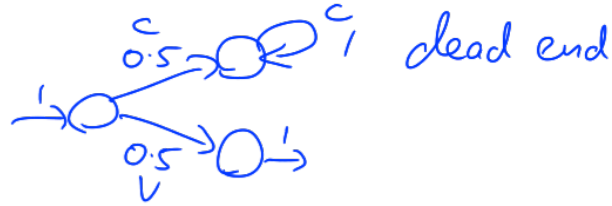
$0 \times \Delta(3, V, 3) = 0 \times 0$

$\sum_w \Pr(w) = 1$

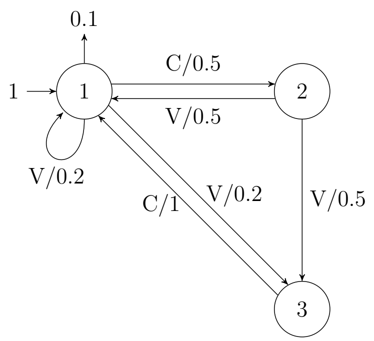
$\Pr(w) \geq 0$

$f(\varepsilon) = 1$
$f(a) = 2$
$f(aa) = 4$
$\vdots$

(24) A *probabilistic finite-state automaton* (PFSA) is a WFSA where:
- all starting weights sum to one;
- each state's outgoing transition weights and ending weight sum to one; and
- there are no "dead ends", i.e. no states from which it is impossible to end.



Here's an example of a PFSA, based on (8) from above.

(25)


$I(1) + I(2) + I(3) = 1$

With a PFSA, we can take the *maximum* across paths, as before, to find the highest probability associated with any analysis of the given string.[1] Or, we can take the *sum* across paths to find the total probability of the string being generated (by *any* analysis)!

$\wedge$	$\vee$	Boolean
$\times$	max	$\mathbb{R}_{\geq 0}$
$\times$	$+$	$[0,1]$

(26)

$$\begin{aligned} \text{for 'VCV' via path [1,1,2,1]:} \quad & 1 \times 0.2 \times 0.5 \times 0.5 \times 0.1 = 0.005 \\ \text{for 'VCV' via path [1,3,1,1]:} \quad & 1 \times 0.2 \times 1 \times 0.2 \times 0.1 = 0.004 \\ \max(0.005, 0.004) &= 0.005 \\ 0.005 + 0.004 &= 0.009 \end{aligned}$$

(27)

$$\begin{aligned} \text{for 'CVCV' via path [1,2,1,2,1]:} \quad & 1 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.1 = 0.00625 \\ \text{for 'CVCV' via path [1,2,3,1,1]:} \quad & 1 \times 0.5 \times 0.5 \times 1 \times 0.2 \times 0.1 = 0.005 \\ \max(0.00625, 0.005) &= 0.00625 \\ 0.00625 + 0.005 &= 0.01125 \end{aligned}$$

$x_1 \quad x_2$

$q_0 \quad q_1 \quad q_2$

$$\bigvee_{q_0} \bigvee_{q_1} \bigvee_{q_2} \left[I(q_0) \wedge \Delta(q_0, x_1, q_1) \wedge \Delta(q_1, x_2, q_2) \wedge F(q_2) \right]$$

$$\max_{q_0} \max_{q_1} \max_{q_2} \left[I(q_0) \times \ldots \times F(q_2) \right]$$

$$\sum_{q_0} \sum_{q_1} \sum_{q_2} \left[I(q_0) \times \ldots \times F(q_2) \right]$$

And the same tricks for using forward (or backward) values also work for calculating these total probabilities, because just like (20) above, *multiplication distributes over addition*:

(28)

$$(a \times k) + (b \times k) = (a + b) \times k \qquad \sum_{x \in S} \Big(g(x) \times k \Big) = \Big(\sum_{x \in S} g(x) \Big) \times k$$

[1] In the context of a probabilistic grammar, the recursive calculation of these probabilities in a table like (18) above is known as the "Viterbi algorithm" — invented by Andrew Viterbi, right here at UCLA in 1967! See `https://en.wikipedia.org/wiki/Andrew_Viterbi`. This was one of many independently-discovered algorithms that were eventually unified under the general perspective presented here.

6 The general pattern: Semirings

	Possible values	Generalized "and" ($\otimes$) and neutral element		Generalized "or" ($\oplus$) and neutral element	
Is it generated?	$\{\mathsf{True}, \mathsf{False}\}$	$\wedge$	True	$\vee$	False
Highest weight?	$\mathbb{R}_{\geq 0}$	$\times$	1	max	0
Total weight?	$\mathbb{R}_{\geq 0}$	$\times$	1	$+$	0

False $\vee$ x = x
max(0, x) = x
0 + x = x

These are all instances of a single algebraic concept.

along a path

(29) A set R, along with an associated "generalized and" operation $\otimes$ and an associated "generalized or" operation $\oplus$, counts as a *semiring* iff the following conditions are satisfied for all $x, y, z \in R$:

combine paths

a. $x \otimes (y \otimes z) = (x \otimes y) \otimes z$
b. There is a particular element $\top \in R$ such that $\top \otimes x = x \otimes \top = x$.
c. $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
d. There is a particular element $\bot \in R$ such that $\bot \oplus x = x \oplus \bot = x$.
e. $x \oplus y = y \oplus x$
f. $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ and $(y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x)$
g. $x \otimes \bot = \bot \otimes x = \bot$

Officially, the semiring is the five-tuple $\mathcal{R} = (R, \oplus, \bot, \otimes, \top)$.

Side note: In the literature, you'll often see the symbol $\otimes$ used where I'm using $\otimes$, and the symbol $\oplus$ used where I'm using $\oplus$. (And in such cases you'll typically see something like 0 or $\overline{0}$ used as the neutral element for $\oplus$, and something like 1 or $\overline{1}$ as the neutral element for $\otimes$.) The choice of $\otimes$ and $\oplus$ is natural if you think of numbers as the "canonical" case of a semiring, and think of others as generalizations of those. I've chosen $\otimes$ and $\oplus$ because this seemed to make more sense given that we, in effect, saw the boolean semiring first and then generalized to others from there. But the important thing is that *what makes a system a semiring is the fact that its pieces interact in the ways stated in* (29), not the names or symbols we use for the pieces of the system.

So here's the general notion of a *semiring-weighted FSA*.

(30) For any semiring $\mathcal{R} = (R, \oplus, \bot, \otimes, \top)$, an $\mathcal{R}$-weighted finite-state automaton is a five-tuple $(Q, \Sigma, I, F, \Delta)$ where:

- Q is a finite set of states;
- Σ, the alphabet, is a finite set of symbols;
- $I : Q \to R$ is a function assigning starting values to states;
- $F : Q \to R$ is a function assigning ending values to states;
- $\Delta : (Q \times \Sigma \times Q) \to R$ is a function assigning values to transitions.

Then for any $\mathcal{R}$-weighted FSA $M = (Q, \Sigma, I, F, \Delta)$, the function $f_M : \Sigma^* \to R$ is defined as:

(31) $$f_M(x_1 \dots x_n) = \bigoplus_{q_0 \in Q} \bigoplus_{q_1 \in Q} \dots \bigoplus_{q_{n-1} \in Q} \bigoplus_{q_n \in Q} \Big[I(q_0) \otimes \Delta(q_0, x_1, q_1) \otimes \dots \otimes \Delta(q_{n-1}, x_n, q_n) \otimes F(q_n) \Big]$$

And the algebraic properties of the semiring operations (especially (29f); recall (21) and (22)) ensure that the now-familiar tricks for calculating these values efficiently will work:

(32) $$\text{fwd}_M(\epsilon)(q) = I(q)$$
$$\text{fwd}_M(x_1 \dots x_n)(q) = \bigoplus_{q_{n-1} \in Q} \Big[\text{fwd}_M(x_1 \dots x_{n-1})(q_{n-1}) \otimes \Delta(q_{n-1}, x_n, q) \Big]$$

(33)
$$\begin{aligned}\text{bwd}_M(\epsilon)(q) &= F(q)\\ \text{bwd}_M(x_1 \dots x_n)(q) &= \bigvee_{q_1 \in Q} \Big[\Delta(q, x_1, q_1) \oslash \text{bwd}_M(x_2 \dots x_n)(q_1)\Big]\end{aligned}$$

(34)
$$f_M(uv) = \bigvee_{q \in Q} \Big[\text{fwd}_M(u)(q) \oslash \text{bwd}_M(v)(q)\Big]$$

(35)
$$f_M(w) = \bigvee_{q \in Q} \Big[\text{fwd}_M(w)(q) \oslash \text{bwd}_M(\epsilon)(q)\Big] = \bigvee_{q \in Q} \Big[\text{fwd}_M(w)(q) \oslash F(q)\Big]$$

(36)
$$f_M(w) = \bigvee_{q \in Q} \Big[\text{fwd}_M(\epsilon)(q) \oslash \text{bwd}_M(w)(q)\Big] = \bigvee_{q \in Q} \Big[I(q) \oslash \text{bwd}_M(w)(q)\Big]$$

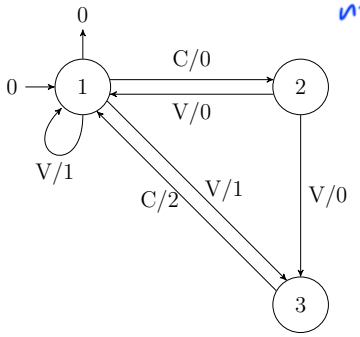
7 Two more useful semirings

7.1 Costs

We can associate a natural number "cost" with each transition.

The numbers in (37) assign a cost of 1 for having an onset, and assign a cost of 2 for not having a coda.

(37)



Here we take the *sum* of the costs along a single path, and take the *minimum* across the various possible paths.

(38)
$$\begin{aligned}\text{for 'VCV' via path [1,1,2,1]:} \quad & 0+1+0+0+0 = 1\\ \text{for 'VCV' via path [1,3,1,1]:} \quad & 0+1+2+1+0 = 4\\ f_M(\text{VCV}) &= \min(1, 4) = 1\end{aligned}$$

If we also take all the transitions not shown in the diagram to have the special value ∞, where

- $x + \infty = \infty + x = \infty$, and
- $\min(x, \infty) = \min(\infty, x) = x$,

then:

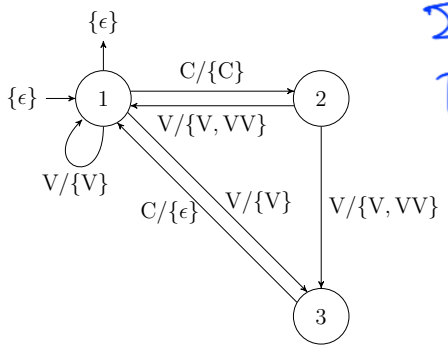
- Each of the "bad paths" that we did not consider in (38) has the value ∞ (because $x + \infty = \infty$).
- So we can consider $f_M(\text{VCV})$ to actually be the minimum over *all* paths; including all the ∞ paths has no effect (because $\min(x, \infty) = x$).
- And therefore if, for some string w, all paths are "bad paths", then $f_M(w) = \infty$.

7.2 Output strings

We can associate a **set of output strings**, over some alphabet Γ, with each transition. (This set of output strings might often be a singleton set.)

The output strings in (39) have the effect of optionally lengthening vowels that follow an onset, and deleting all codas.

(39)



Here we use *(lifted) concatenation* to combine weights along a single path and take the *union* across the various possible paths.

(40) $$X \bullet Y = \{u +\!\!+ v \mid u \in X, v \in Y\}$$

(41) for 'VCV' via path [1,1,2,1]: $\{\epsilon\} \bullet \{\text{V}\} \bullet \{\text{C}\} \bullet \{\text{V, VV}\} \bullet \{\epsilon\} = \{\text{VCV, VCVV}\}$
for 'VCV' via path [1,3,1,1]: $\{\epsilon\} \bullet \{\text{V}\} \bullet \{\epsilon\} \bullet \{\text{V}\} \bullet \{\epsilon\} = \{\text{VV}\}$

$$f_M(\text{VCV}) = \{\text{VCV, VCVV}\} \cup \{\text{VV}\} = \{\text{VCV, VCVV, VV}\}$$

Plus, if we take all the transitions not shown in the diagram to have the value $\{\}$ then:

- Each of the "bad paths" that we did not consider in (41) has the value $\{\}$ (because $X \bullet \{\} = \{\}$).
- So we can consider $f_M(\text{VCV})$ to actually be the union over *all* paths; including all the $\{\}$ paths has no effect (because $X \cup \{\} = X$).
- And therefore if, for some string w, all paths are "bad paths", then $f_M(w) = \{\}$.

	Possible values	Generalized "and" ($\otimes$) and neutral element		Generalized "or" ($\oplus$) and neutral element	
Is it generated?	{True, False}	$\wedge$	True	$\vee$	False
Highest weight?	$\mathbb{R}_{\geq 0}$	$\times$	1	max	0
Total weight?	$\mathbb{R}_{\geq 0}$	$\times$	1	$+$	0
Lowest cost?	$\mathbb{N} \cup \{\infty\}$	$+$	0	min	∞
Output strings?	$\mathcal{P}(\Gamma^*)$	$\bullet$	$\{\epsilon\}$	$\cup$	$\{\}$