Thursday – August 29, 2024

Semiring & Variants of formal grammars

- true or false

1 Reminder/summary: Boolean FSAs

Generation of strings is fundamentally defined like this:

- Initial - transitions

(1) $$x_1x_2...x_n \in \mathcal{L}(M) \Leftrightarrow \bigvee_{q_0\in Q}\bigvee_{q_1\in Q}...\bigvee_{q_n\in Q}\Big[I(q_0)\wedge\Delta(q_0,x_1,q_1)\wedge ...\wedge\Delta(q_{n-1},x_n,q_n)\wedge F(q_n)\Big]$$

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

(2) a. $fwd_M(\epsilon)(q) = I(q)$ - Base case (is it initial)

b. $$fwd_M(x_1...x_n)(q) = \bigvee_{q_{n-1}\in Q}\Big[fwd_M(x_1...x_{n-1})(q_{n-1})\wedge\Delta(q_{n-1},x_n,q)\Big]$$

(3) a. $bwd_M(\epsilon)(q) = F(q)$ - is it final

b. $$bwd_M(x_1...x_n)(q) = \bigvee_{q_1\in Q}\Big[\Delta(q,x_1,q_1)\wedge bwd_M(x_2...x_n)(q_1)\Big]$$

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

- split up into two strings

(4) a. $$uv \in \mathcal{L}(M) \Leftrightarrow \bigvee_{q\in Q}\Big[fwd_M(u)(q)\wedge bwd_M(v)(q)\Big]$$

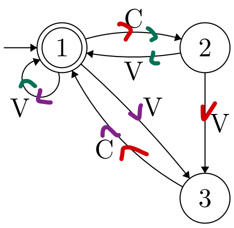
(language of machine

b. $$w \in \mathcal{L}(M) \Leftrightarrow \bigvee_{q\in Q}\Big[fwd_M(w)(q)\wedge bwd_M(\epsilon)(q)\Big] \quad\Leftrightarrow \bigvee_{q\in Q}\Big[fwd_M(w)(q)\wedge F(q)\Big]$$

c. $$w \in \mathcal{L}(M) \Leftrightarrow \bigvee_{q\in Q}\Big[fwd_M(\epsilon)(q)\wedge bwd_M(w)(q)\Big] \quad\Leftrightarrow \bigvee_{q\in Q}\Big[I(q)\wedge bwd_M(w)(q)\Big]$$

Notice that everything here is about computing with *Booleans*, using *conjunction and disjunction.* Take a look at the FSA in (5). What does this FSA say about the string 'CVC'? What about the string 'VCV'?

true true

(5) 

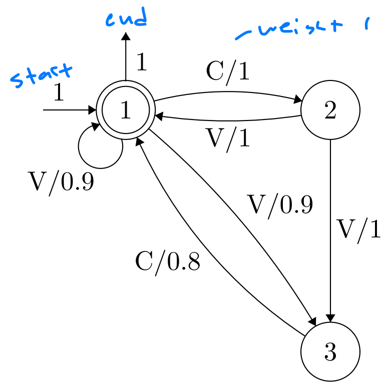
- In both cases what it says is just "yes"/"True"— even though they are generated in different ways, 'VCV' is generated *two* ways and 'CVC' only one way, etc.
- As soon as one choice of q_0, q_1, q_2, q_3 gives us a "True," all the other choices of states make no difference.

2 Introducing weights

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

arbitratry (not probabilities)

(6)



You can think of this weighted FSA as implementing the following linguistic idea:

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

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

(7) a. $f_M(\text{CV}) = \overset{\text{start}}{1} \times \overset{\text{trans.}}{1} \times \overset{\text{trans.}}{1} \times \overset{\text{end}}{1}$
b. $f_M(\text{VC}) = 1 \times 0.9 \times 0.8 \times 1$
c. $f_M(\text{CVC}) = 1 \times 1 \times 1 \times 0.8 \times 1$
d. $f_M(\text{CVV}) = 1 \times 1 \times 1 \times 0.9 \times 1$
e. $f_M(\text{VVC}) = 1 \times 0.9 \times 0.9 \times 0.8 \times 1$

For "$\overset{\text{multiple}}{\text{ambiguous}}$" strings, we multiply along each path and take the *maximum*. The idea is that a string is as good as its best analysis.

(8) 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]: $1 \times 0.9 \times 0.8 \times 0.9 \times 1 = 0.648$
$f_M(\text{VCV}) = \max(\underline{0.9}, 0.648) = 0.9$

(9) 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]: $1 \times 1 \times 1 \times 0.8 \times 0.9 \times 1 = 0.72$
$f_M(\text{CVCV}) = \max(\underline{1}, 0.72) = 1$

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 don't affect the overall value assigned to a string, because $\max(x, 0) = x$.

(10) 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]: $1 \times 1 \times 0 \times 1 = 0$ no path from final to final using C
$f_M(\text{VC}) = \max(0.72, 0) = 0.72$

Question: what would happen if every weight in a particular weighted FSA were either zero or one?

- ambiguity would not matter
· we would get a regular as boolean FSA again

3 Weighted FSAs, formally

Now for the careful definitions.

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

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

Nonnegative real num

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

switch or for max

(12)
$$f_M(x_1 ... x_n) = \max_{\text{all possible paths } p} \Big[\text{value for } x_1 ... x_n \text{ via path } p\Big]$$
$$= \max_{q_0 \in Q} \max_{q_1 \in Q} \dots \max_{q_{n-1} \in Q} \max_{q_n \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \cdots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n)\Big]$$

multiplication

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

(13)
$$f_M(\epsilon) = \max_{q_0 \in Q} \Big[I(q_0) \times F(q_0)\Big]$$

The definition in (12) makes the task of calculating $f_M(w)$ for some long string w 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 $fwd_M(w)(q)$ as the best product-of-weights that 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.

across all paths, which is best path

(14) $$fwd_M(x_1 ... x_n)(q) = \max_{q_0 \in Q} \max_{q_1 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q) \Big]$$

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

, multiply

(15) $$f_M(w) = \max_{q_n \in Q} \Big[fwd_M(w)(q_n) \times F(q_n) \Big]$$

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

- The calculation of $fwd_M(w)(q_n)$ 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:

(16) Using the WFSA in (6):

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

' what is starting weight

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:

(17) a. $fwd_M(\epsilon)(q) = I(q)$

b. $$fwd_M(x_1 ... x_n)(q) = \max_{q_{n-1} \in Q} \Big[fwd_M(x_1 ... x_{n-1})(q_{n-1}) \times \Delta(q_{n-1}, x_n, q) \Big]$$

don't need

It may take a bit of thought to convince yourself that this works, but the important logic is the same as for (15) above:

- The calculation of $fwd_M(x_1 ... x_{n-1})(q_{n-1})$ already took into account *all* of 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:

(18) a. $\max(a \times k, b \times k) = \max(a, b) \times k$

b. $$\max_{x \in S} \Big(g(x) \times k \Big) = \Big(\max_{x \in S} (g(x)) \Big) \times k$$

$\max(2 \times 3, 4 \times 3) = \max(2,4) \times 3$

This is what justifies the following algebraic reshuffles, which get us to the equation in (15) by "pulling out" $F(q_n)$:

(19) $f_M(x_1 ... x_n)$

$$\begin{aligned}
&= \max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \max_{q_n \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n) \Big] \\
&= \max_{q_n \in Q} \Big[\max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \times F(q_n) \Big] \Big] \\
&= \max_{q_n \in Q} \Big[\max_{q_0 \in Q} \dots \max_{q_{n-1} \in Q} \Big[I(q_0) \times \Delta(q_0, x_1, q_1) \times \dots \times \Delta(q_{n-1}, x_n, q_n) \Big] \times F(q_n) \Big] \\
&= \max_{q_n \in Q} \Big[fwd_M(x_1 ... x_n)(q_n) \times F(q_n) \Big]
\end{aligned}$$

Similarly, we can get to the equation in (17) by "pulling out" $\Delta(q_{n-1}, x_n, q)$:

(20) $fwd_M(x_1 ... x_n)(q)$

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

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 (6) above), add up the weights of all generated strings and divide:

(21) a. $Z(M) = \sum_{w \in \Sigma^*} f_M(w)$

b. $P(w) = \dfrac{f_M(w)}{Z(M)}$

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

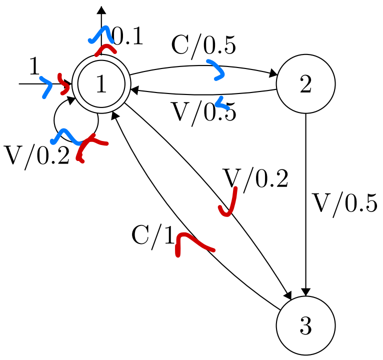
A more common approach is to set up a WFSA so that it obeys certain conditions which guarantee that the sum $Z(M)$ turns out to be 1, so that $f_M(w)$ just is the probability of w. A WFSA that satisfies these conditions is called a **probabilistic finite-state automaton**.

(22) A probabilistic finite-state automaton (PFSA) is WFSA where:

a. all starting weights sum to one;
b. each state's outgoing transition weights and ending weight sum to one; and
c. 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 (6) from above.

(23)



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)!

(24) For 'VCV' via path [1,1,2,1]: $1 \times 0.2 \times 0.5 \times 0.5 \times 0.1 = 0.005$
For 'VCV' via path [1,3,1,1]: $1 \times 0.2 \times 1 \times 0.2 \times 0.1 = 0.004$
$\max(0.005, 0.004) = 0.005$
$0.005 + 0.004 = 0.009$

(25) For 'CVCV' via path [1,2,1,2,1]: $1 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.1 = 0.00625$
For 'CVCV' via path [1,2,3,1,1]: $1 \times 0.5 \times 0.5 \times 1 \times 0.2 \times 0.1 = 0.005$
$\max(0.00625, 0.005) = 0.00625$
$0.00625 + 0.005 = 0.01125$

And the same tricks for using forward (or backward) values also work for calculating these total probabilities, because just like (18) above, multiplication distributes over addition:

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

6 The general pattern: Semirings

True ∧ x = x

False ∨ x = x

	Possible values	Generalized "and" ($\oslash$)	Neutral element ($\top$)	Generalized "or" ($\oslash$)	Neutral element ($\bot$)
boolean Is it generated?	{True, False}	$\wedge$	True	$\vee$	False
weighted Highest weight?	$\mathbb{R}_{\geq 0}$ non-neg. real numbers	$\times$	1 - 1×y = y	max	0 - max(0,x) = x
Total weight?	$\mathbb{R}_{\geq 0}$	$\times$	1	+	0 - 0 + x = x

These are all instances of a single algebraic concept.

[1] In the context of a probabilistic grammar, the recursive calculation of these probabilities in a table like (16) 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.

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

- $x \owedge (y \owedge z) = (x \owedge y) \owedge z$
- There is a particular element $\top \in R$ such that $\top \owedge x = x \owedge \top = x$
- $x \ovee (y \ovee z) = (x \ovee y) \ovee z$
- There is a particular element $\bot \in R$ such that $\bot \ovee x = x \ovee \bot = x$
- $x \ovee y = y \ovee x$
- $x \owedge (y \ovee z) = (x \owedge y) \ovee (x \owedge z)$ and $(y \ovee z) \owedge x = (y \owedge x) \ovee (z \owedge x)$
- $x \owedge \bot = \bot \owedge x = \bot$

- general FSA

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

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

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

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

(28)
$$f_M(x_1 ... x_n) = \bigovee_{q_0 \in Q} \bigovee_{q_1 \in Q} \cdots \bigovee_{q_{n-1} \in Q} \bigovee_{q_n \in Q} \Big[I(q_0) \owedge \Delta(q_0, x_1, q_1) \owedge ... \owedge \Delta(q_{n-1}, x_n, q_n) \owedge F(q_n) \Big]$$

And the algebraic properties of the semiring operations (especially the distributivity properties) ensure that the now-familiar tricks for calculating these values efficiently will work:

(29)
a. $fwd_M(\epsilon)(q) = I(q)$ - Initial state
b. $fwd_M(x_1 ... x_n)(q) = \bigovee_{q_{n-1} \in Q} \Big[fwd_M(x_1 ... x_{n-1})(q_{n-1}) \owedge \Delta(q_{n-1}, x_n, q) \Big]$

- placeholders for what we use

(30)
a. $bwd_M(\epsilon)(q) = F(q)$
b. $bwd_M(x_1 ... x_n)(q) = \bigovee_{q_1 \in Q} \Big[\Delta(q, x_1, q_1) \owedge bwd_M(x_2 ... x_n)(q_1) \Big]$

(31)
a. $f_M(uv) = \bigovee_{q \in Q} \Big[fwd_M(u)(q) \owedge bwd_M(v)(q) \Big]$

b. $f_M(w) = \bigovee_{q \in Q} \Big[fwd_M(w)(q) \owedge bwd_M(\epsilon)(q) \Big] = \bigovee_{q \in Q} \Big[fwd_M(w)(q) \owedge F(q) \Big]$

c. $f_M(w) = \bigovee_{q \in Q} \Big[fwd_M(\epsilon)(q) \owedge bwd_M(w)(q) \Big] = \bigovee_{q \in Q} \Big[I(q) \owedge bwd_M(w)(q) \Big]$

A semiring helps us construct a general FSA

7 Two more useful semirings

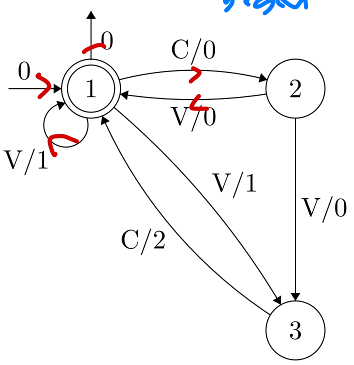
7.1 Costs

We can associate a natural number "cost" with each transition. For example, the numbers in (32) assign a cost of 1 for not having an onset, and assign a cost of 2 for having a coda.

begin

end

(32)



note min

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

(33) For 'VCV' via path [1,1,2,1]: $0 + 1 + 0 + 0 + 0 = 1$
For 'VCV' via path [1,3,1,1]: $0 + 1 + 2 + 1 + 0 = 4$
$f_M(\text{VCV}) = \min(1, 4) = 1$ - total cost of 1

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 (33) has the value ∞ (because $x + \infty = \infty$).
- So we can consider $f_M(\text{VCV})$ to actually be the minimum over *all possible paths*— including all of the ∞ paths has no effect (because $\min(x, \infty) = x$).
- And therefore if, for some string w, all paths are "bad paths," then $f_M(x) = \infty$.

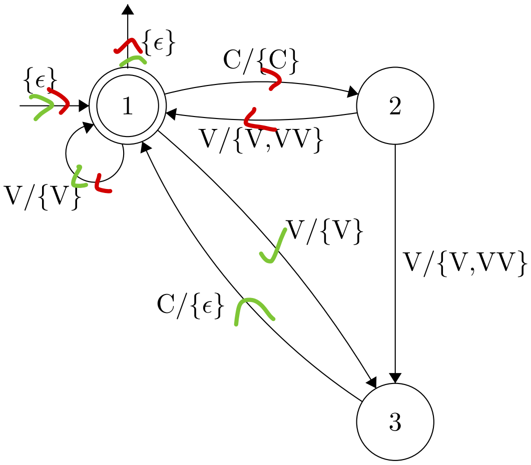
cost

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.)

For instance, the output strings in (34) have the effect of optionally lengthening vowels that follow an onset, and deleting all codas.

(34)



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

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

(36) For 'VCV' via path [1,1,2,1]: $\{\epsilon\} \cdot \{V\} \cdot \{C\} \cdot \{V,VV\} \cdot \{\epsilon\} = \{VCV,VCVV\}$
For 'VCV' via path [1,3,1,1]: $\{\epsilon\} \cdot \{V\} \cdot \{\epsilon\} \cdot \{V\} \cdot \{\epsilon\} = \{VV\}$
$f_M(VCV) = \{VCV,VCVV\} \cup \{VV\} = \{VCV,VCVV,VV\}$ ~ outputs

And if we take all of the transitions not shown in the diagram to have the value {}, then: (nothing to concatenate

- Each of the "bad paths" that we did not consider in (36) has the value {} (because $X \cdot \{\} = \{\}$).
- So we can consider f_M(VCV) to actually be the union over *all* possible paths— including all of the {} paths has no effect (because $X \cup \{\} = X$).
- And therefore if, for some string w, all paths are "bad paths," then $f_M(x) = \{\}$.

7.3 Semiring-weighted FSAs, summarized

	Possible values	Generalized "and" ($\owedge$)	Neutral element ($\top$)	Generalized "or" ($\ovee$)	Neutral element ($\bot$)
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?	$\mathcal{P}(\Gamma^*)$	$\cdot$	$\{\epsilon\}$	$\cup$	{}