LECTURE - 3

Recall from last lecture: A weighted automata with n states over a semiring $\mathbb{S}$ and alphabet Σ can be represented as $\langle \lambda, \mu, \gamma \rangle$

where λ is a $1 \times n$ row vector

γ is a $n \times 1$ column vector

μ_a is a $n \times n$ matrix

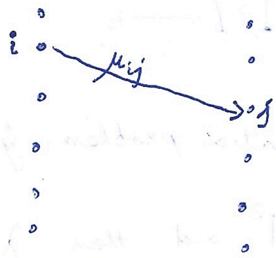
- this is the weight of the transition on a. If not present, consider it to be 0.

Suppose $u \in \Sigma^*$. Then, $\mu(u)_{ij}$ [as defined in the last lecture] is the sum of weights of all paths from i to j on u.

Note that

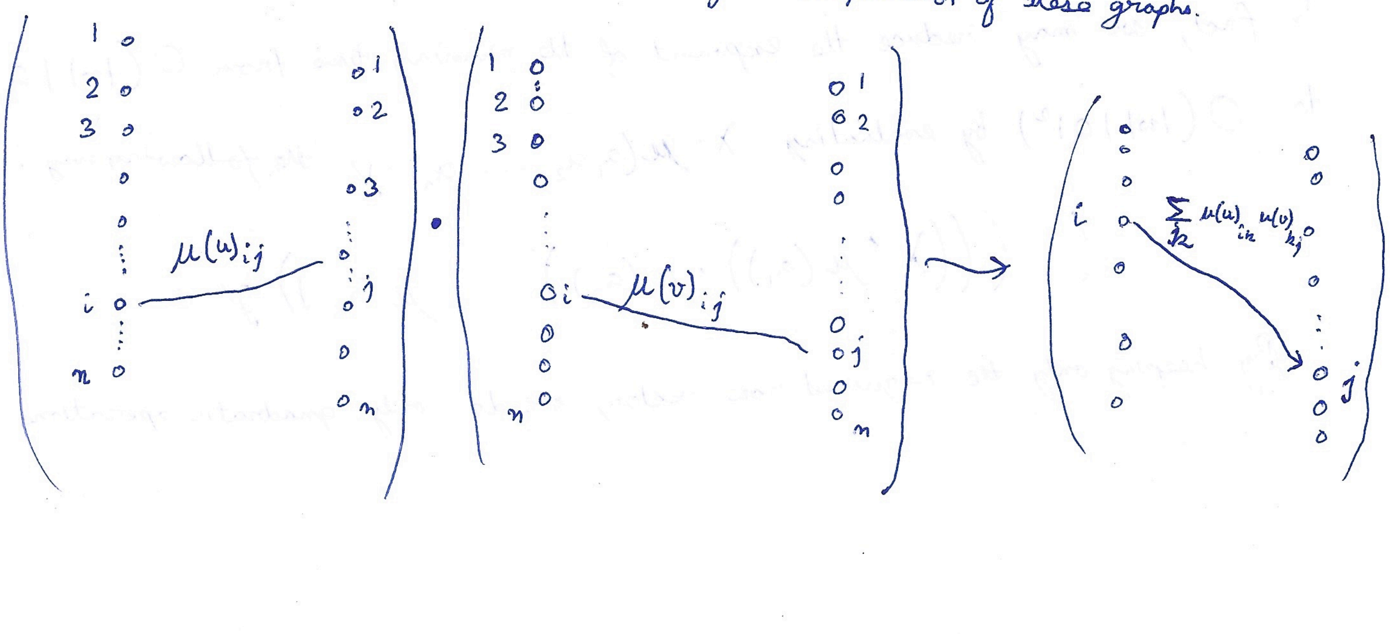
$$\mu(u) \underset{\text{matrix multiplication}}{\cdot} \mu(v) = \mu(u \underset{\text{concat}}{\cdot} v)$$

One may also think of $\mu(u)$ as a bipartite graph where the ~~weight~~ edge from Q_i to Q_j



would be $\mu(u)_{ij}$ which is an abstraction of all the paths from i to j on u together.

Viewed this way, matrix multiplication is just composition of these graphs.



Given w, you can compute $\mu(w)$ by multiplying matrices.

The "initial weight" vector λ takes a linear combination of the rows in $\mu(u)$.

$$\underbrace{\begin{array}{ccccccc} \multicolumn{7}{c}{\lambda} \\ [\,0 & 0 & a & \cdots & b & \cdots & 0\,] \\ & & \uparrow & & \uparrow & & \\ & & i & & j & & \end{array}} \quad \begin{array}{c} \\ R_i \\ \\ R_j \end{array} \overset{\mu(w)}{\begin{bmatrix} \\ \text{———} \\ \text{———} \\ \\ \text{———} \\ \text{———} \\ \end{bmatrix}} = a R_i + b R_j$$

Now, the "final ~~weights~~" vector again computes a linear combination of the components.

$$\begin{array}{ccccccc} \multicolumn{7}{c}{\lambda\,\mu(u)} \\ [\,\cdots & a & \cdots & b & \cdots & c & \cdots\,] \\ & \uparrow & & \uparrow & & \uparrow & \\ & i & & j & & k & \end{array} \quad \overset{\gamma}{\begin{bmatrix} \vdots \\ \alpha \\ \vdots \\ \beta \\ \vdots \\ \gamma \end{bmatrix}} \begin{array}{l} \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \leftarrow k \end{array} = \alpha a + b\beta + \gamma c$$

This gives an algorithm for the evaluation problem. Given $\langle \lambda, \mu, \gamma \rangle$ and $w \in \Sigma^*$, we may compute $\mu(w)$ in time $|w| |Q|^3$ and then $\lambda \cdot \mu(w) \cdot \gamma$.

For a fixed automaton, this is actually linear in time

In fact, we may reduce the exponent of the running time from $O(|w| |Q|^3)$ to $O(|w| |Q|^2)$ by evaluating $\lambda \cdot \mu(a_1 a_2 \cdots a_n) \cdot \gamma$ the following way:

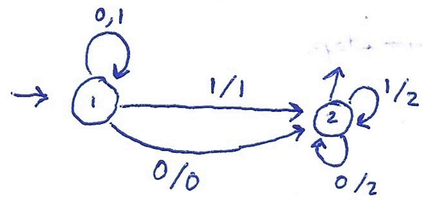
$$(\ldots(((\lambda \cdot \mu(a_1)) \cdot \mu(a_2)) \ldots \mu(a_n)) \gamma$$

By keeping only the required row vector, we do only quadratic operations everytime.

This representations may be useful to prove claims about the function $[\![A]\!]$ rigorously.

Example

Let $b_n b_{n-1} \cdots b_1 b_0$ be a sequence of binary digits. Then, $(b_n \cdots b_0)_2$ is defined as $\sum_{i=0}^{n} b_i 2^i$. Consider the following automaton:



Any run can be seen as a sequence of states q_1 ~~and~~ followed by a sequence of q_2's. If there are $(n-i)$ q_1's and (i) q_2's, then the weight of the corresponding run is 2^i if $b_i = 1$ and 0 otherwise.

This shows that the automaton computes the represented number.

A formal proof of this would be to look at the representation as follows:

$$\lambda = [1 \quad 0] \qquad \gamma = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\mu_0 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \qquad \mu_1 = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}$$

Claim: If $w = b_n b_{n-1} \cdots b_0$, then $\mu(w) = \begin{pmatrix} 1 & \text{bin}(w) \\ 0 & 2^{|w|} \end{pmatrix}$

Proof By induction.

Base Case True for $w = 1$ and $w = 0$

Induction Step

Suppose this is true for w.

Then, we may check that $\mu(w0) = \mu(w)\mu(0) = \begin{pmatrix} 1 & \text{bin}(w) \\ 0 & 2^{|w|} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$

$$= \begin{pmatrix} 1 & 2 \cdot \text{bin}(w) \\ 0 & 2 \cdot 2^{|w|} \end{pmatrix} = \begin{pmatrix} 1 & \text{bin}(w0) \\ 0 & 2^{|w0|} \end{pmatrix}$$

$$\mu(w1) = \begin{pmatrix} 1 & bin(w) \\ 0 & 2^{|w|} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2\,bin(w)+1 \\ 0 & 2 \cdot 2^{|w|} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & Bin(w1) \\ 0 & 2^{|w1|} \end{pmatrix}$$

This proves the inductive ~~claim~~ step.

QED

Support Languages

We wish to use weighted automata as an intermediate device for defining qualitative languages. Supports are one such mechanism.

Given A, an automata, we have $[\![A]\!] : \Sigma^* \longrightarrow S$

Define $\text{Support}([\![A]\!]) = \{ w \mid [\![A]\!](w) \neq 0 \}$

Can a support language be non-regular?

Yes! Consider $\{ w \in \Sigma^* \mid |w|_a \neq |w|_b \}$

This is the support of the automaton discussed in LECTURE-2 that computes $|w|_a - |w|_b$ over $(\mathbb{Z}, +, \times)$

Can a support language be non-context-free?

Yes! Consider $\boxed{\{ w \in \{a,b,c\}^* \mid |w|_a \neq |w|_b \text{ or } |w|_a \neq |w|_c \}}$

this is context-free!

Algorithmic Question: Given A, is the support language of A non-empty?

Partial Solution

Defn A semiring S is 0-sum free if $\forall a, b \in S, a, b \neq 0 \Rightarrow a+b \neq 0$

On semirings which are 0-sum free, the existence of a path with no 0 edges from ~~an initial~~ state with non-zero initial weight to a state with non-zero final weight is sufficient.

This formulation of the non-emptyness problem over 0-sum free semirings puts it in NLOGSPACE.

This is a direct generalization of the approach for checking non-emptyness of standard NFA. This approach works since the boolean semiring $(\mathbb{B}, \wedge, \vee)$ is 0-sum free. Another example of a 0-sum free semiring is $(\mathbb{N} \cup \{+\infty\}, \min, +)$.