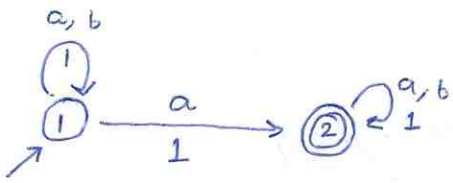
LECTURE - 1

Σ - finite alphabet

Language : $\Sigma^* \longrightarrow \{0, 1\}$

Weighted Language : $\Sigma^* \longrightarrow \mathbb{R}$

weight domain



Transitions labelled by letter <u>and a weight</u>

What runs are there for a b b a ?

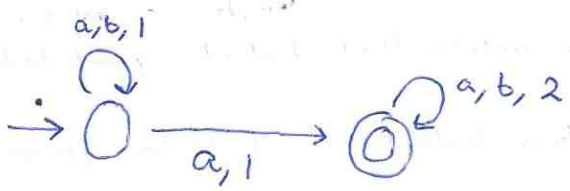
ρ_1 states: 1 1 1 1 2 $\longrightarrow$ 1

ρ_2 states 1 2 2 2 2 $\longrightarrow$ 1

product of weights

Take sum of product of weights. So, $[\![A]\!] (a\,b\,b\,a) = 2$

The automata above counts the number of a's



Convention : Not Mentioning an weight means 1.

What does this automaton compute?

Answer : the number represented by a, b in binary if a is 1, b is 0

The weights come from a semiring:

A semiring is a set S equipped with operations $+$ and $\times$ and distinguished elements 0 and 1 such that

- $(S, +, 0)$ is a commutative monoid
- $(S, \times, 1)$ is a monoid
- $\times$ distributes over $+$
- $a \cdot 0 = 0 \cdot a = 0$

This is often written as $(S, +, \times, 0, 1)$.

Caution: - Inverses may not exist
- $\times$ may not commute.

<u>Examples</u>

1. $(\mathbb{N}, +, \times, 0, 1)$
2. $(\mathbb{R}, +, \times, 0, 1)$
3. $(\mathbb{B}, \vee, \wedge, 0, 1)$ — standard NFA are just weighted automata over this semiring
4. $(\{\infty\} \cup \mathbb{N}, \min, +, +\infty, 0)$ — tropical semiring
5. $(2^{\Sigma^*}, \cup, \cdot, \{\}, \{\varepsilon\})$

We could also generalise initial and final states. Simply multiply an initial and a final weight at the states through which the run leaves and enters. (We use labelled arrows to show this)

<u>Exercise</u> Construct an automaton that computes the length of the minimal a block. Use the tropical semiring.

Simplifying Assumption: Assume your word is of the form $b\Sigma^*b$

LECTURE - 2

Weighted Language - functions of type $\Sigma^* \longrightarrow \mathbb{S}$

$$\left[\mathbb{S} = \left(\begin{matrix} (S, +, \cdot, 0, 1) \\ \text{- semiring} \end{matrix} \right) \right]$$

Weighted Automata over $\underset{\text{alphabet}}{\Sigma}$ and $\underset{\text{semiring}}{\mathbb{S}}$

Syntax:

$$\mathcal{A} = (Q, \Delta, \lambda, \gamma, \mu)$$

Q - finite set of "states"

$\Delta \subseteq Q \times \Sigma \times Q$

edge weights $\left\{ \begin{array}{l} \mu : \Delta \longrightarrow S \\ \text{[or could directly specify } \Delta \text{ as } Q \times \Sigma \times Q \longrightarrow S \\ \qquad \text{with } 0 \text{ on absent edges]} \end{array} \right.$

λ - initial weight : $Q \longrightarrow S$

γ - final weight : $Q \longrightarrow S$

Semantics:

Given $w = a_1 a_2 \cdots a_n \in \Sigma^*$

define $$[\![A]\!](w) := \sum_{\substack{q_0 \cdots q_n \\ \in Q^n}} \lambda(q_0) \left(\prod_{i=1}^{n} \mu(q_{i-1}, a_i, q_i) \right) \gamma(q_n)$$

$[\![A]\!] : \Sigma^* \longrightarrow \mathbb{S}$ — this is the weighted language

Exercises

1. Construct an weighted automata that counts the number of a's in a word
   Take $\Sigma = \{a, b\}$, $\mathbb{S} = (\mathbb{Z}, +, \cdot, 0, 1)$

2. Construct an automata that computes the same function over $(\mathbb{Z} \cup \{+\infty\}, \min, +, +\infty, 0)$

3. " " " " " $w \mapsto |w|_a - |w_b|$ in $(\mathbb{Z}, +, \cdot)$

Evaluation Problem

Input : A over $\mathbb{S}$, and $w \in \Sigma^*$
Output : $[\![A]\!](w)$

First definition is $|Q|^{|w|}$. Redefine the automata using matrices.
Represent A by $\langle \lambda, \mu(a), \mu(b), \gamma \rangle$

where $\lambda, \gamma \in \mathbb{S}^{|Q|}$, as before

and $\mu(a)$ is a matrix over $\mathbb{S}$ s.t

$\mu(a)_{i,j} = x$ iff there is a transition from i to j on a with wt. x.

Now, extend μ to Σ^* ~~with~~ in the following way:

$$\mu(\varepsilon) = I$$

$$\mu(wa) = \mu(w) \cdot \mu(a)$$ matrix multiplication

$$\mu(wb) = \mu(w) \cdot \mu(b)$$

~~Define~~ With this definition,

<u>Claim</u> $\mu(w)_{pq}$ = Sum of Product of Weights over all Paths from p to q on w

<u>Proof</u> <u>Base Case</u> $\mu(\varepsilon)$, $\mu(a)$, $\mu(b)$

Follows from definition.

<u>Inductive Case</u>

$$\mu(wa) = \mu(w) \cdot \mu(a)$$

$$\Longrightarrow \mu(wa)_{pq} = \sum_r \mu(w)_{pr} \cdot \mu(a)_{rq}$$

$$= \sum_r \begin{pmatrix} \text{Sum of weights} \\ \text{of } w\text{-paths from} \\ p \text{ to } r \end{pmatrix} \begin{pmatrix} \text{Weight of} \\ \text{edge from } r \text{ to } q \end{pmatrix} \quad [\text{IH}]$$

$$= \sum_r \begin{pmatrix} \text{Sum of weights of } wa \text{ paths that} \\ \text{pass through } r \text{ on reading } w \end{pmatrix} \quad [\text{distributivity}]$$

$$= \text{Sum of weights of all } wa \text{ paths}$$

$\square$

Intuition: $\mu(a)$, $\mu(b)$ are actions that ~~distort a probability distribution~~ transform a weight distribution

λ is an ~~ini~~ initial distribution. γ is an aggregation function.

Observe that for an word $w = a_1 a_2 \cdots a_n$ and an automata as above, we can compute

$$[\![A]\!](w) = \lambda \cdot \mu(w) \cdot \gamma$$

in time $O(|w|\,|Q|^3 \beta)$ by doing matrix multiplication.

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

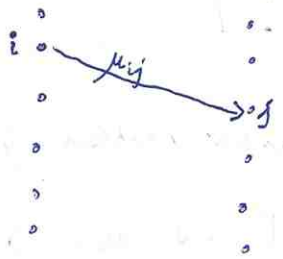
- this is the weight of the transition on a. If not present, consider it to be 0.

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.

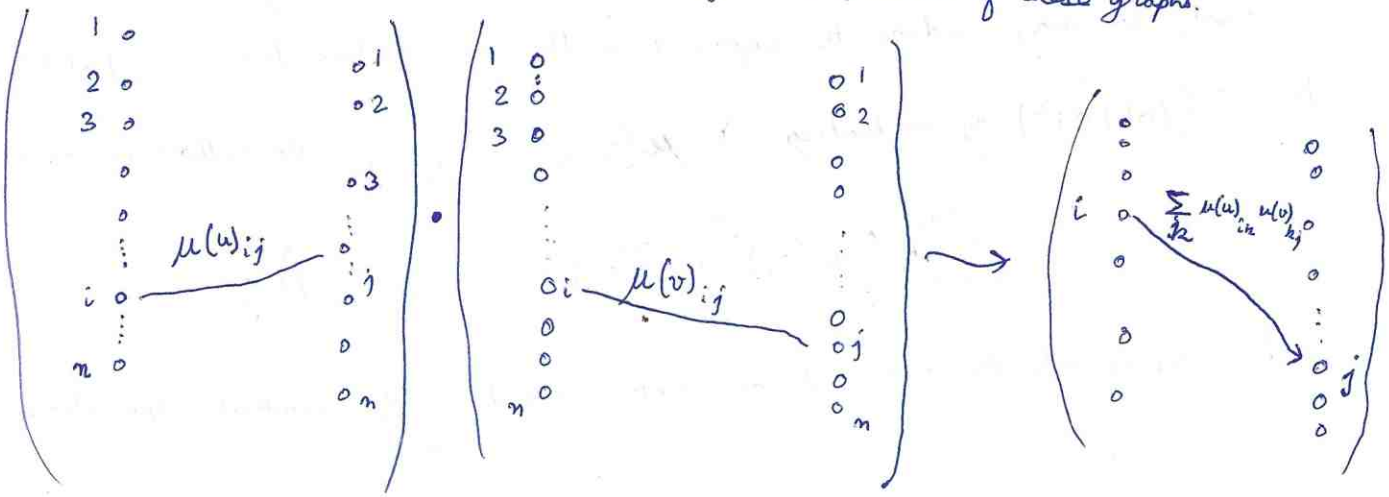
Note that

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

One may also think of $\mu(u)$ as a bipartite graph where the ~~arrow~~ 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 $\mu(w)$.

$$\underset{\substack{\uparrow \\ i}}{\lambda} \quad \underset{\substack{\uparrow \\ j}}{\mu(w)}$$

$$\begin{array}{c} \\ [0\ 0\ a \cdots b \cdots 0] \end{array} \begin{array}{c} \\ R_i \\ \\ R_j \end{array} \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} = a R_i + b R_j$$

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

$$\lambda \quad \mu(w) \qquad \gamma$$

$$[\cdots a \cdots b \cdots c \cdots] \begin{bmatrix} \vdots \\ \alpha \\ \vdots \\ \beta \\ \vdots \\ \gamma \end{bmatrix} \begin{array}{l} \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \leftarrow k \end{array} = \alpha a + b\beta + \gamma c$$

$\uparrow i \quad \uparrow j \quad \uparrow k$

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:

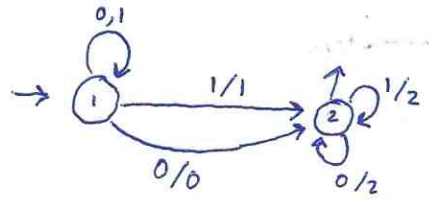
$$(\cdots(((\lambda \cdot \mu(a_1)) \cdot \mu(a_2)) \cdots \mu(a_n))\gamma$$

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

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

<u>Example</u>

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

<u>Proof</u> By induction.

<u>Base Case</u> True for $w = 1$ and $w = 0$

<u>Induction Step</u>

Suppose this is true for w.

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

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

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

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

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 $\mathcal{A}$, 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 $\mathcal{A}$, is the support language of $\mathcal{A}$ 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 semering $(\mathbb{B}, \wedge, \vee)$ is 0-sum free. Another example of a 0-sum free semiring is $(\mathbb{N} \cup \{+\infty\}, \min, +)$.

LECTURE - 4

Recall : Given an weighted automaton $\mathcal{A}$, Support $([\![A]\!]) = \{ w \mid [\![A]\!](w) \neq 0 \}$

Non-emptiness Problem for Support Languages :

Input : Weighted Automaton $\mathcal{A}$ over $\mathbb{S}$ and Σ

Question : Is Support $([\![A]\!]) \neq \phi$?

We return to the matrix representation for $\mathcal{A} = \langle \lambda, \mu, \gamma \rangle$

λ - $1 \times n$ row vector $\in \mathbb{S}^n$

μ - $\Sigma \longrightarrow$ ($n \times n$ ~~row~~ matrix over $\mathbb{S}$)

γ - $n \times 1$ column vector

Can think of μ_a for $a \in \Sigma$ as a weight distortion.

With each word, the automaton $\mathcal{A}$ associates a vector in $\mathbb{S}^n$, namely the vector $\lambda \cdot \mu(w)$ with the word w. Here, $\mu(w)$ is defined as in LECTURE-2.

For the sake of ~~simpli~~ the results that will follow, we will assume that $\mathbb{S}$ is a field, i.e, inverses of $+$ and $\times$ exist in $\mathbb{S}$.

Define the set of <u>Reachable Vectors</u> in $\mathbb{S}^n$ as follows :

$$\lambda \mu (\Sigma^*) = \left\{ \lambda \mu(w) \mid w \in \Sigma^* \right\} \subseteq \mathbb{S}^n$$

We are interested in the span of reachable vectors in $\mathbb{S}^n$, denoted $\langle \lambda \cdot \mu(\Sigma^*) \rangle$.

This is a subspace of $\mathbb{S}^n$ and we may compute its basis.

The following algorithm computes ~~the~~ a basis for $\langle \lambda \mu(\Sigma^*) \rangle$

$B \leftarrow \{\lambda\}$

Todo $\leftarrow \{\lambda\}$

while Todo $\neq \phi$:

  remove x from Todo

  If $x \cdot \mu(a) \notin \langle B \rangle$,

    add $x \mu(a)$ to B

    add $x \mu(a)$ to Todo

  If $x \cdot \mu(b) \notin \langle B \rangle$,

    add $x \mu(b)$ to B

    add $x \mu(b)$ to Todo

Output B

Now, we argue the correctness of the algorithm above.

- We observe that the set B is always linearly independent. This can be established as a loop invariant. This proves that B would be linearly independent if the algorithm halts

- Since B is always linearly independent, there may be at most n additions to Todo. After another n iterations, Todo must be empty. Therefore, the algorithm terminates.

- Suppose the algorithm returns $B = \{x_1 \dots x_k\}$. We argue, by induction, that $\lambda \mu(w) \in \langle B \rangle$, for all $w \in \Sigma^*$

  + <u>Base Case</u> $\lambda \mu(\varepsilon) = \lambda$ is added to B manually in the beginning

  + <u>Inductive Step</u> Suppose $\lambda \mu(w) \in \langle B \rangle$ and in particular $\lambda \cdot \mu(w) = \sum_{i=1}^{k} \alpha_i x_i$ for $\alpha_i \in S$.

$$(\lambda \cdot \mu(w)) \cdot \mu(a) = \sum_{i=1}^{k} \alpha_i x_i \cdot \mu(a)$$

But the algorithm ~~is such~~ maintains that $x_i = \lambda\mu(w_i)$ for some $|w_i| < n$

So, $$(\lambda\mu(w)) \cdot \mu(a) = \sum_{i=1}^{k} \alpha_i (x_i \, \mu(a))$$

$$= \sum_{i=1}^{k} \alpha_i \lambda\mu(w_i a) \qquad \text{—— (1)}$$

Now, either $\lambda\mu(w_i a) \in B$, in which case we are done. Otherwise $\lambda\mu(w_i a) \notin B$. But we know that $\lambda\mu(w_i) \in B$. So, $\lambda\mu(w_i)$ was an element of Todo at some point. But $\lambda\mu(w_i a)$ was not added to B. This means $\lambda\mu(w_i a) = \sum_{i=1}^{k} \beta_i x_i$

So, (1) can be rewritten as $$\sum_{i=1}^{k} \alpha_i \left(\sum_{i=1}^{k} \beta_i x_i\right)$$

but this sum can be expressed as a ~~sum~~ linear combination of elements in B.

A similar argument holds for $\lambda\mu(wb)$.

QED

Checking whether a given vector is in the span of B can be done using Gaussian Elimination which requires $O(n^3)$ operations.

Thus, the algorithm terminates in $O(|\Sigma| \times n \times n^3)$ time.

<u>Claim</u> Suppose $B = \{x_1 \dots x_k\}$ is a basis of $\langle \lambda\mu(\Sigma^*) \rangle$ obtained by the algorithm above. Then $\mathrm{Support}(\llbracket \langle \lambda, \mu, \gamma \rangle \rrbracket)$ is empty iff $x_i \cdot \gamma = 0$ for each i.

<u>Proof</u> ($\Leftarrow$) Given any word $w \in \Sigma^*$, $\lambda\mu(w)$ can be realised as a linear combination of B, say $\sum_{i=1}^{k} x_i \alpha_i$. Therefore, $\lambda\mu(w)\gamma$ is $\sum \alpha_i x_i \gamma_i$

$$= \sum \alpha_i (x_i \cdot \gamma_i) = \sum \alpha_i \cdot 0 = 0$$

Thus $w \notin \mathrm{Supp}(\llbracket \langle \lambda, \mu, \gamma \rangle \rrbracket)$

($\Longrightarrow$) If there is some x_i s.t $x_i \cdot \gamma \neq 0$, there is a corresponding w_i s.t $\lambda \mu(w_i) = x_i$ and hence $\lambda \mu(w_i) \gamma \neq 0$. Thus, $w_i \in Supp([\![\langle \lambda, \mu, \gamma \rangle]\!])$

QED

So, we have a polynomial time algorithm for checking the non-emptiness of the support language.

Equivalence Problem for weighted Automata

Input : $\mathcal{A}_1$, $\mathcal{A}_2$ - weighted automata

Question : Is $[\![A_1]\!] = [\![A_2]\!]$?

Observe that if $\mathcal{A}_1 = \langle \lambda_1, \mu_1, \gamma_1 \rangle$ and $\mathcal{A}_2 = \langle \lambda_2, \mu_2 \gamma_2 \rangle$, we may consider

$\mathcal{A} = \langle \lambda, \mu, \gamma \rangle$ with

$$\lambda = [\overbrace{\cdots}^{\lambda_1} \quad \underbrace{\cdots}_{-\lambda_2}]$$

$$\mu(a) = \begin{bmatrix} \mu_1(a) & 0 \\ 0 & \mu_2(a) \end{bmatrix} \quad \text{for } a \in \Sigma$$

$$\gamma = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}$$

This is simply the disjoint union of $\mathcal{A}_1$ and $\mathcal{A}_2$ with negated initial weights for $\mathcal{A}_2$. $[\![\mathcal{A}]\!]$ then computes $[\![\mathcal{A}_1]\!] - [\![\mathcal{A}_2]\!]$.

Thus, to answer the equivalence problem, we may construct $\mathcal{A}$ as above and check if Support $([\![\mathcal{A}]\!]) = \phi$.

Exercise Assume that $\mathbb{S}$ is an ordered field. We say that $[\![\mathcal{A}_1]\!] \leq [\![\mathcal{A}_2]\!]$ if $[\![\mathcal{A}_1]\!](w) \leq [\![\mathcal{A}_2]\!](w)$, read as, $\mathcal{A}_1$ is dominated by $\mathcal{A}_2$. Given $\mathcal{A}_1$ and $\mathcal{A}_2$, check algorithmically if $\mathcal{A}_1$ dominates $\mathcal{A}_2$.

<u>Remark</u> The arguments above hold if S is embedded in some field. For example, $\mathbb{N}$ can be embedded in $\mathbb{Q}$ and so can be $\mathbb{Z}$.

In general, this may not be possible. For example, consider $(\mathbb{N} \cup \{\infty\}, \min, +)$

Suppose we could embed this in a ring with elements n' for each $n \in \mathbb{N}$ s.t. $\min(n, n') = \infty$. Then, we could get a contradiction as follows:

$$\min(1, 2) = \min(1, 3)$$
$$\Rightarrow \min(1', \min(1, 2)) = \min(1', \min(1, 3))$$
$$\Rightarrow \min(\min(1', 1), 2) = \min(\min(1', 1), 3)$$
$$\Rightarrow \min(\infty, 2) = \min(\infty, 3)$$
$$\Rightarrow 2 = 3$$

Thus, in general, cancellativity is necessary for a semiring to be embedded in a field. In the semiring $(\{0, 1\}, \vee, \wedge)$, we have $0 \vee 1 = 1 \vee 1 = 1$. Hence, these arguments do not hold. Indeed, checking equivalence for NFA is PSPACE-complete.

However, if a semiring can be embedded in a ring without any zero-divisors, (i.e, the ring maintains $m \cdot n = 0 \Rightarrow m = 0 \vee n = 0$), then it may be embedded in its corresponding field of fractions.

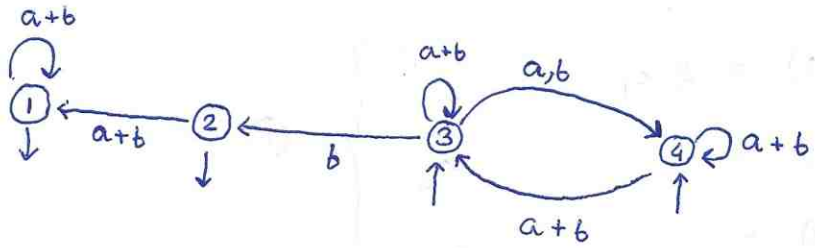
LECTURE - 5

Recall: We defined Reachable Vectors $\lambda\mu(\Sigma^*)$ last time and gave an algorithm to compute a basis for $\langle \lambda\mu(\Sigma^*)\rangle$

Apriori we have that $\langle \lambda\mu(\Sigma^*)\rangle \subseteq \mathbb{S}^n$, but we may be able to find a basis $\{x_1, \dots x_k\}$ for $\langle \lambda\mu(\Sigma^*)\rangle$ with $k \leq n$. In other words, the set of reachable vectors may be obtained as a k dimensional subspace of $\mathbb{S}^n$. In such a case, we should be able to represent the given automaton ~~as slame~~ with k states. Indeed, this is what we shall argue. (We assume throughout that $\mathbb{S}$ is a field.)

<u>Example</u>

$\mathcal{A} =$



$$\lambda = [0 \quad 0 \quad 1 \quad 1]$$

$$\mu(a) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \mu(b) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

By running the algorithm from LECTURE - 4 with minor modifications, we see that the following is indeed a basis for $\langle \lambda\mu(\Sigma^*)\rangle$:

$$\text{Basis} = \{x_1, x_2, x_3\}$$

$x_1 = \lambda = (0\ 0\ 1\ 1)$

$x_2 = (0\ 1\ 0\ 0)$

$x_3 = (1\ 0\ 0\ 0)$

Define $X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

The weighted automaton may be understood as an object that assigns with each word $w \in \Sigma^*$, the vector $\lambda\mu(w) \in \mathbb{S}^n$. However, if we indeed find that $\langle \lambda\mu(w) \rangle \simeq \mathbb{S}^k$, for some $k < n$, then we may represent each $\lambda\mu(w) \in \mathbb{S}^n$ via $\lambda'\mu'(w) \in \mathbb{S}^k$ such that there is a (linear) transformation that takes $\lambda'\mu'(w)$ to $\lambda\mu(w)$. This transform happens to be X as we will check.

In summary, if $(\alpha_1 \ \alpha_2 \ \alpha_3)$ is a vector with respect to the new basis, then $(\alpha_1 \ \alpha_2 \ \alpha_3)\, X$ is the coordinates of the same vector with respect to the standard basis.

~~Suppose that~~

Now, returning to the example, we have that:

$$\left.\begin{aligned} x_1\mu(a) &= 2x_1 \\ x_2\mu(a) &= x_3 \\ x_3\mu(a) &= x_3 \end{aligned}\right\} : \ (1)$$

and

$$\left.\begin{aligned} x_1\mu(b) &= 2x_1 + x_2 \\ x_2\mu(b) &= x_3 \\ x_3\mu(b) &= x_3 \end{aligned}\right\} \ (2)$$

So, define

$$\mu'(a) := \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mu'(b) := \begin{pmatrix} 2 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Note that System (1) and System (2) are simply facts that

$$X\mu(a) = \mu'(a)X \qquad \text{and} \qquad X\mu(b) = \mu'(b)X$$

These equations are just manifestations of the fact that we wish the following diagram to commute

$$\begin{tikzcd} S^n \arrow[r, "\mu(a)"] & S^n \\ S^k \arrow[u, "X"] \arrow[r, "\mu'(a)"'] & S^k \arrow[u, "X"'] \end{tikzcd}$$

Its not hard to argue that we will be able to find such a $\mu'(a)$ everytime.

Indeed, suppose $x_1 \in \langle \lambda \mu(\Sigma^*) \rangle$, then there are words $w_1 \dots w_k$ such that $x_1 = \sum_1^k \alpha_i w_i$. Then, we have that $x_1 \mu(a) = (\sum \alpha_i w_i) \mu(a)$

$$= \sum \alpha_i \mu(w_i a) \in \langle \lambda \mu(\Sigma^*) \rangle$$

Since $\{x_1 \dots x_k\}$ is a basis, we should be able to represent $x_1 \mu(a)$ as a linear combination of elements in them. Using a similar argument, we may find λ' s.t. $\lambda' X = \lambda$.

Finally, define $\gamma' := X\gamma$. Again, note that if $(\alpha_1\ \alpha_2\ \alpha_3)$ is a representation of a vector in the new basis, then $(\alpha_1\ \alpha_2\ \alpha_3) X$ is their representation in the standard basis and $\big((\alpha_1\ \alpha_2\ \alpha_3) X\big)\gamma = (\alpha_1\ \alpha_2\ \alpha_3)\gamma'$

Thus, we may define $\mathcal{A}'$ whose states are $\{x_1, x_2, x_3\}$ and transitions and other data are given as follows: $\mathcal{A}' = \langle \lambda', \mu', \gamma' \rangle$

<u>Claim</u> For all $w \in \Sigma^*$, $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$

<u>Proof</u> The claim is equivalent to the fact that $\lambda' \mu'(w) \gamma' = \lambda \mu(w) \gamma$

$$\iff \lambda' \mu'(w) X \gamma = \lambda \mu(w) \gamma$$

We can prove this equivalent statement by induction on w. $\iff \lambda' \mu'(w) X = \lambda \mu(w)$

<u>Base Case</u> If $w = \varepsilon$, Then, we need to check that $\lambda' X = \lambda$

This is true by definition.

<u>Inductive Step</u>

$$\begin{aligned}
\lambda' \cdot \mu'(wa) \cdot X &= \lambda' \mu'(w) \mu'(a) X \\
&= \lambda' \mu'(w) X \mu(a) && \text{[definition of } \mu'(a)\text{]} \\
&= \lambda \mu(w) \mu(a) && \text{[Induction Hypothesis]} \\
&= \lambda \mu(wa)
\end{aligned}$$

A similar argument holds for $\lambda' \cdot \mu'(wb) \cdot X = \lambda \mu(wb)$

QED

We call this automata the <u>Reduced Automata</u>. This construction is ~~indeed~~ a counterpart to the subset construction, in the sense that we club together ~~different runs~~ different configurations as linear combinations of a certain basis. Note that the field structure is necessary here since, the coefficients of the reduced automata may contain additive or multiplicative inverses. This is the reason we cannot do such a construction for NFA, as the "reachable vectors" do not admit a small basis.

<u>Exercise</u> (Not necessarily relevant to the above theory)

Given an NFA $\mathcal{A}$, show that

Determerize (Reverse (Determerize (Reverse $\mathcal{A}$)))

is ~~a~~ the minimal DFA for $\mathcal{A}$

LECTURE - 6

Recall - last time we saw reduction of weighted automata which was equivalent to subset construction.

Theorem If $\mathcal{A} = \langle \lambda, \mu, \gamma \rangle$ is an weighted automaton over $\mathbb{S}$ and $\mathbb{S}$ can be embedded in a field, then if $\mathcal{A}' =$ ~~Det ∘ Rev ∘ Det~~

$$(\text{Reduce} \circ \text{Transpose} \circ \text{Reduce} \circ \text{Transpose})(\mathcal{A})$$

then $[\![\mathcal{A}']\!] = [\![\mathcal{A}]\!]$ and for any other $\mathcal{B}$ with $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$, $|\mathcal{B}| \geq |\mathcal{A}'|$ where $|\mathcal{B}|$ denotes the number of states in $\mathcal{B}$.

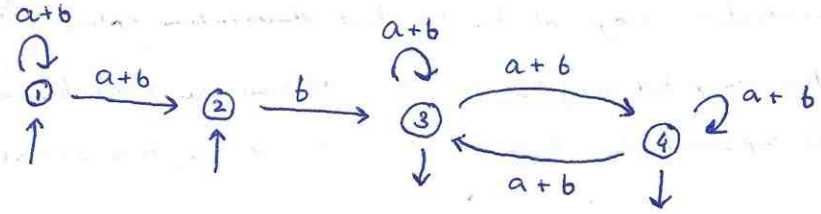
Try proving this

$\boxtimes$

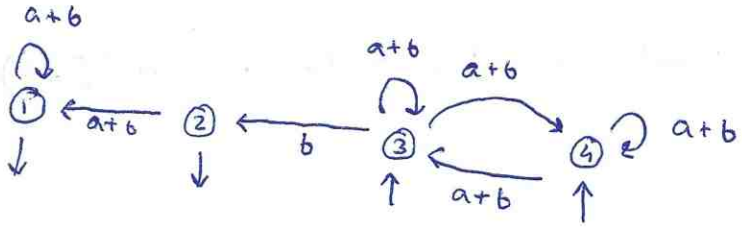
By transpose $(\mathcal{A})$, we mean the automaton $\langle \gamma, \mu^T, \lambda \rangle$ which is an analogue for the reversal operation.

Example

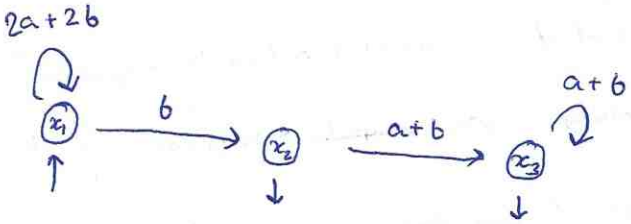
$\mathcal{A} :=$



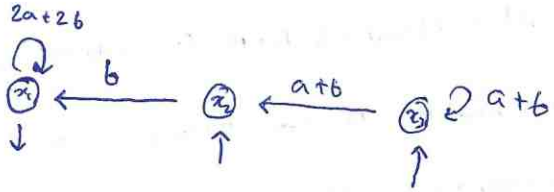
Transpose $(\mathcal{A})$ =



Reduce (Transpose $(\mathcal{A})$) =



Transpose (Reduce (Transpose $(\mathcal{A})$)) =



We reduce the last automaton further

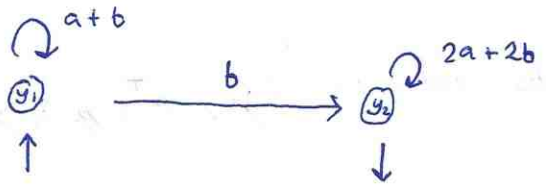
$$\lambda = y_1 = (0 \quad 1 \quad 1)$$

$$y_1 \cdot \mu(a) = (0 \quad 1 \quad 1) = y_1$$

$$y_1 \cdot \mu(b) = (1 \quad 1 \quad 1) = \underbrace{(1 \quad 0 \quad 0)}_{y_2} + (0 \quad 1 \quad 1)$$
$$= y_2 + y_1$$

$$Y = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\gamma' = Y\gamma = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \gamma'$$



(Minimal Automaton)

<u>Aside</u> State space minimization may not be the best ~~description~~ notion. For example, even though the function computed may be via a 2-dimensional reachable set of vectors, it may still be representable as a finite union of 1-dimensional objects as well.

A Roadmap for the Course

If $\mathcal{A}$ is an weighted automaton over $\mathbb{S}$, we defined $\mathrm{Supp}([\![\mathcal{A}]\!])$.

1) Support Languages

- Regularity

  Is the support of an weighted automata regular?

  Not necessarily. Can ~~compute~~ "recognise" $\{a^n b^n \mid n \in \mathbb{N}\}$ using $(\mathbb{Z}, +, \times)$.

  When $\mathbb{S}$ is 0-sum free, $\mathrm{Support}([\![\mathcal{A}]\!])$ is regular. (Erase all weights on $\mathcal{A}$ to get a classical NFA for support $([\![\mathcal{A}]\!])$)

  This problem is non-trivial in a general setting

- Non-emptiness Algorithm

  We know how to do this for semirings inside fields as well as 0-sum free semirings.

2) Equivalence

- We saw how to do this when S embeds inside a field. Polytime.
- For usual NFA, test two inclusions. May take exponential time.
- Undecidable for tropical Semirings. (Krob).

3) Minimization

4) Threshold Languages

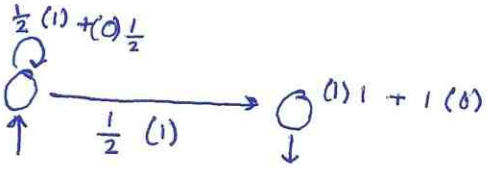
$$L_{\bowtie k} = \{ w \mid [\![A]\!](w) \bowtie k \}$$

$$\bowtie \in \{ =, \neq, <, >, \leq, \geq \}$$

usually this is of general interest

Threshold Languages may not be recursive (or even recursively enumerable) in general.

Consider the following automaton over $\mathbb{R}$.



Given $a_1 a_2 a_3 \cdots a_n$, this computes $\sum_{i=1}^{n} a_i 2^{-i}$

Let α, β be reals with $0 < \alpha < \beta < 1$.

Since dyadic rationals are dense in $\mathbb{R}$, there must be some element in $L_{\geq \beta}$ which is not in $L_{\geq \alpha}$. So, there are uncountably many languages $L_{\geq k}$ but only countably many are ~~recognizing~~ recursive.

Later, we will see the following: If α is an isolated cut point, i.e, $\exists \varepsilon > 0$ s.t. $|x - \alpha| < \varepsilon \Rightarrow \nexists w$ s.t $f(w) = x$, then $L_{\geq \alpha}$ is regular

5) Dominance Problem

- The Boundedness Problem may be easier

(Given $\mathcal{A}$, does $[\![\mathcal{A}]\!](w) < k$, for some k, for all w?)

6) Power Series

Regular Expression counterpart for weighted languages.

Representing weights of ~~language~~ words via coefficients of power series.

LECTURE - 7

Tropical Semirings (Trop)

$(\mathbb{Z} \cup \{+\infty\}, \min, +, +\infty, 0)$

or $(\mathbb{N} \cup \{+\infty\}), \min, +, +\infty, 0)$

Weighted Automaton over Trop are called distance automata.

We commented that Support Languages over Trop are regular. However, many other problems over Trop are undecidable. Today, we consider the following problem.

Suppose $\bowtie \in \{=, \neq, \leq, <\}$ and T is a tropical semiring.

Input : $\mathcal{A}, \mathcal{B}$ - weighted automata over T

Qn : Is $[\![\mathcal{A}]\!] \bowtie [\![\mathcal{B}]\!]$? i.e, for all w, does $[\![\mathcal{A}]\!](w) \bowtie [\![\mathcal{B}]\!](w)$?

We first restrict our attention to $T = \mathbb{Z}$ and $\bowtie = \leq$

<u>Minsky Machines</u> (2-counter automaton)

A minsky machine is a pair (Q, Δ) where Q is a finite set of states and

$\Delta \subseteq Q \times Ops \times Q$

where $Ops = \{c_1 = 0$
$c_2 = 0$
$c_1 ++$
$c_2 --$
$c_1 --$
$c_2 ++\}$

2-counter automata here are considered with the usual semantics.

Recall that the following problem for 2-counter automata is undecidable.

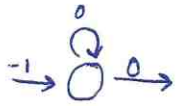
Input : $(Q, \Delta), q_0, q_{target}$

Qn : $\exists\, x_1, x_2$ s.t $(q_0, 0, 0) \xrightarrow{*} (q_{target}, x_1, x_2)$?

Given a counter automata (Q, Δ) and q_0, q_{target}, we will show an effective reduction to an automaton $\mathcal{A}$ over $(\mathbb{Z}, \min, +)$ s.t. $[\![\mathcal{A}]\!](w) \leq -1$ iff q_{target} is not reachable.

Note that it is easy to construct $\mathcal{B}$ s.t. for all $w \in \Sigma^*$, $[\![\mathcal{B}]\!](w) = -1$.

Take $\mathcal{B}$ to be



Thus, this reduction would show that the problem at hand is undecidable.

<u>The Reduction</u>

Let $\Sigma = \{a, b\} \cup \Delta$. Reducing this to a binary alphabet is a straightforward (but tedious) exercise.

We will represent a run $(q_0, \ell_1, k_1) \xrightarrow{t_1} (q_1, \ell_2, k_2) \xrightarrow{t_2} \cdots \xrightarrow{t_{n-1}} (q_n, \ell_n, k_n)$ as

$$a^{\ell_1} b^{k_1} t_1 \, a^{\ell_2} b^{k_2} t_2 \cdots t_{n-1} \, a^{\ell_n} b^{k_n}$$

We intend to produce $\mathcal{A}$ such that $[\![\mathcal{A}]\!](w) = 0$ iff w encodes a legal run of the machine that ~~takes~~ reaches q_{target} and $[\![\mathcal{A}]\!](w) \leq -1$ otherwise.

We characterize a non-"good" run using properties 1 through 11 and design weighted automata $\mathcal{A}_1, \mathcal{A}_2, \cdots \mathcal{A}_{11}$ s.t. $\mathcal{A}_i(w) \leq -1$ iff w satisfies property i and 0 otherwise. We define $\mathcal{A}$ to be the disjoint union of $\mathcal{A}_1 \cdots \mathcal{A}_{11}$. It is clear that $[\![\mathcal{A}]\!](w) \leq -1$ iff $[\![\mathcal{A}_i]\!](w) \leq -1$ for some i. We list the first five properties below:

1) w is not of the form $(a^* b^*)(\Delta a^* b^*)^*$
2) w does not start in q_0
3) w does not end in q_{target}
4) w encodes incoherent consecutive transitions
5) w takes a transition with a 0-test even though the corresponding counter is not zero.

We note that the first five properties are regular.

If R is a regular language, we may express R via an weighted automata $\langle \lambda, \mu, \gamma \rangle$ where λ and γ encode initial and final states of the corresponding DFA and μ places 0 on transitions which are present and $+\infty$ on those which are non-present.

Given such a $\langle \lambda, \mu, \gamma \rangle$, we may consider $\mathcal{A}_i = \langle \lambda', \mu', \gamma' \rangle \boxplus \langle \lambda'', \mu'', \gamma'' \rangle$

where λ' places a (-1) on the initial state, μ' is μ and γ' is γ

and $\langle \lambda'', \mu'', \gamma'' \rangle$ is some automaton that has weight θ on every word.

~~Now~~ $\mathcal{A}_i$ has the desired ~~prop~~ properties for $i \in \{1, \dots 5\}$.

Now, we move to the sixth property:

6) The run takes a wrong increment on c_1, i.e, it encodes a move of the form

$$(q_i, x_1, x_2) \xrightarrow{c_1 ++} (q_j, x_3, x_4) \text{ where } x_3 \neq x_1 + 1.$$

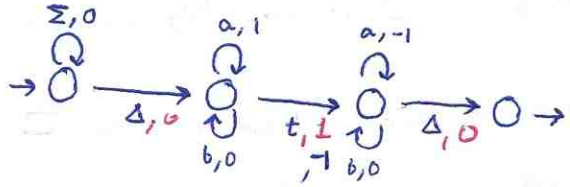
This means, either, (i) $x_3 > x_1 + 1$

or (ii) $x_3 < x_1 + 1$

We shall construct $\mathcal{A}_{6i}$ and $\mathcal{A}_{6ii}$ which will produce a negative weight if condition (i) or (ii) hold. $\mathcal{A}_6$ is the disjoint union of these two automata.

Below, we outline the construction of $\mathcal{A}_{6i}$. The construction of $\mathcal{A}_{6ii}$ is similar.

We wish to detect patterns of the form $a^{x_1} b^* t\, a^{x_3} b^*$ where $x_3 > x_1 + 1$ and output a negative number if such a pattern is found. We take disjoint union of all automata of the following form where t is of the form $c_1 ++$



We can handle the next few properties in a similar fashion:

7) wrong increment on c_2

8) wrong decrement on c_1

9) wrong decrement on c_2

10) If $(q_i, x_1, x_2) \xrightarrow{t} (q_j, x_3, x_4)$ and t does not increment or decrement c_1 but $x_1 \neq x_3$

11) Similarly for ~~x_2 and x_4~~ c_2.

We can handle the cases (10) and (11) by understanding $x_1 \neq x_3$ as $x_1 > x_3$ or $x_1 < x_3$ and using a method similar to 6.

This completes the reduction.

Further Undecidability Results

We saw that checking $\mathcal{A} \leq -1$ was undecidable. Since $\mathcal{A} \leq -1$ is the same as $\mathcal{A} < 0$, the same problem with $<$ substituted for $\bowtie$ is also undecidable.

Checking if $\mathcal{A} = \mathcal{B}$ is also undecidable.

<u>Proof</u> Reduce $\leq$ to $=$

Suppose f and g are two functions $\Sigma^* \to T$ realised by some distance automaton

Then

$$[f \leq g] \text{ iff } [f = \min(f, g)]$$

Now, $\min(f, g)$ can be easily realised by a distance automaton by taking the disjoint union of the automata that computes f and that computes g.

$\square$

This immediately shows that checking $\mathcal{A} \neq \mathcal{B}$ is also undecidable.

The analogous problems are also undecidable over $(\mathbb{N}, \min, +)$

<u>Proof</u> We reduce equality testing on $\mathbb{Z}$ to equality testing on $\mathbb{N}$.

Suppose $\mathcal{A}$ and $\mathcal{B}$ are distance automata over $\mathbb{Z}$. Suppose k is the smallest weight that appears in either of them. Define $\mathcal{A}'$ and $\mathcal{B}'$ to be distance automata where every transition with weight u has been replaced with $u + k$ (where $x + \infty \triangleq \infty$)

$\mathcal{A}'$ and $\mathcal{B}'$ now have weights in $\mathbb{N}$.

Notice that $[\![\mathcal{A}']\!](w) = [\![\mathcal{A}]\!](w) + k(|w| + 2)$ and similarly for $\mathcal{B}$.

Thus,

$$[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!] \text{ iff } [\![\mathcal{A}']\!] = [\![\mathcal{B}']\!]$$

QED

LECTURE - 8

Last time, we saw Distance Automata (word weights are distances).

- Word Evaluation

  linear in $|w|$

  cubic in number of states

- Support Language

  Always regular

- Threshold Problem
- $A \bowtie B$

  } undecidable

- Boundedness Problem

  Are the weights produced by this automaton bounded by some value?

  + Equivalence modulo boundedness

  ~~For any two bounded subsets of Σ^*, is the distance of each word equal to the distance~~

  + Cost Functions
  + B-automata

  } The mathematically oriented reader should look these up

Probabilistic Automata

Weighted Automata over $(\mathbb{R}, +, \times)$ with syntactic restrictions:

As before, a weighted automaton A with $\langle \lambda, \mu, \gamma \rangle$ with restrictions

- all entries in λ or μ or γ come from $[0, 1]$
- $\sum_i \lambda_i = 1$
- $\sum_j \mu^a_{ij} = 1 \quad \forall i \ \forall a \in \Sigma$ — (ii)
- $\gamma_i \in \{0, 1\} \quad \forall i$

These are also called <u>probabilistic reactive systems</u>.

A variation where words are stochastically generated, instead of being fed by the environment is known as <u>generative probabilistic automata</u>. (We will not look at this model)

Condition (ii) above is often stated as $\mu(a)$ being a <u>stochastic matrix</u>, i.e, a matrix whose row sums are 1.

With the definition of $\mu(w)$ as before, we claim that $\mu(w)$ is always a stochastic vector for all $w \in \Sigma^*$.

To verify this, we first check that the product of two stochastic matrices is stochastic.

If (a_{ij}) and (b_{ij}) are stochastic matrices, and (c_{ij}) is their product, then

$$\sum_j c_{ij} = \sum_j \left(\sum_k a_{ik} b_{kj} \right) = \sum_k \sum_j a_{ik} b_{kj}$$

$$= \sum_k a_{ik} \left(\sum_j b_{kj} \right) = \sum_k a_{ik} = 1$$

as desired.

Similarly, we can show that the product of a stochastic vector with a stochastic matrix is also a stochastic vector.

<u>Some Closure Properties</u>

1. If $\mathcal{A}$ is a probabilistic automaton, then there exists another probabilistic automaton $\mathcal{B}$ s.t $[\![\mathcal{B}]\!]$ realises the function $1 - [\![\mathcal{A}]\!]$

<u>Proof</u> If $\mathcal{A} = \langle \lambda, \mu, \gamma \rangle$, consider $\mathcal{B} = \langle \lambda, \mu, \bar{\gamma} \rangle$ where $\bar{\gamma}_i = 1 - \gamma_i$ $\square$

2. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are probabilistic automata, there is a probabilistic automata $\mathcal{B}$ st

$[\![\mathcal{B}]\!] = [\![\mathcal{A}_1]\!] \, [\![\mathcal{A}_2]\!]$, i.e, $\forall w \in \Sigma^*$, $[\![\mathcal{B}]\!](w) = [\![\mathcal{A}_1]\!](w) \times [\![\mathcal{A}_2]\!](w)$

<u>Proof</u> Idea: Construct a Product Automaton.

Let $Q(\mathcal{B}) = Q(\mathcal{A}_1) \times Q(\mathcal{A}_2)$

Then, let $\lambda^{\mathcal{B}}_{(p,q)} = \lambda^{\mathcal{A}_1}_p \cdot \lambda^{\mathcal{A}_2}_q$

$$\mu^{\mathcal{B}}_{(p,q) \to (r,s)}(a) = \mu^{\mathcal{A}_1}_{p \to r}(a) \cdot \mu^{\mathcal{A}_2}_{q \to s}(a)$$

$$\gamma^{\mathcal{B}}_{(p,q)} = \gamma^{\mathcal{A}_1}_p \cdot \gamma^{\mathcal{A}_2}_q$$ (This is intuitively like the $\wedge$ of the two values)

$\square$

3. Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are probabilistic automata, and α, β are in $[0, 1]$ with $\alpha + \beta \leq 1$

Then, there exists $\mathcal{B}$ which realises the function $\alpha [\![\mathcal{A}_1]\!] + \beta [\![\mathcal{A}_2]\!]$

<u>Proof</u>

Let $\perp$ be an extra state not in $\mathcal{A}_1 \cup \mathcal{A}_2$

Construct an automaton $\mathcal{B}$ by taking the disjoint union of scaled copies $\alpha \mathcal{A}_1$ and $\beta \mathcal{A}_2$ as follows:

$$\lambda = [\underbrace{\qquad}_{\alpha \lambda_{\mathcal{A}}} \underbrace{\qquad}_{\beta \lambda_{\mathcal{A}_2}} \underbrace{\quad}_{1-\alpha-\beta}]$$

$$\mu(a) = \begin{bmatrix} \boxed{\alpha \mu^{\mathcal{A}_1}(a)} & & \\ & \boxed{\beta \mu^{\mathcal{A}_2}(a)} & \\ & & \boxed{1} \end{bmatrix}$$

$$\gamma = [\underbrace{\qquad}_{\gamma_{\mathcal{A}_1}} \underbrace{\qquad}_{\gamma_{\mathcal{A}_2}} 0]$$

$\square$

4. If $\mathcal{A}$ is a probabilistic automata, then we can construct $\mathcal{B}$ such that $[\![\mathcal{B}]\!](w) = \begin{cases} 0 & \text{if } w = \varepsilon \\ [\![\mathcal{A}]\!](w) & \text{o/w} \end{cases}$

<u>Proof</u> Add an extra state $\top$

$$\lambda = [\overset{\top}{1} \overbrace{\ldots 0000000 \ldots}^{Q}]$$

$$\mu(a) = \begin{array}{c|c} & \top \quad Q \\ \hline \top & 0 \quad [\lambda^{\mathcal{A}} \cdot \mu^{\mathcal{A}}(a)] \\ Q & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \mu^{\mathcal{A}}(a) \end{array}$$

$$\gamma = (0 \underbrace{\qquad}_{\gamma^{\mathcal{A}}})$$

$\square$

Probabilistic Language

$\mathcal{L} \subseteq \Sigma^*$ is a probabilistic language iff there exists a probabilistic automata $\mathcal{A}$ and an $\alpha \in [0,1]$ s.t $\mathcal{L} = \{ w \mid [\![\mathcal{A}]\!](w) > \alpha \}$.

We shall ~~revisit~~ see undecidability results about probabilistic languages next time.

LECTURE - 9

Last Time: Probabilistic Automata

- Some closure Properties
- Definition of Probabilistic Languages

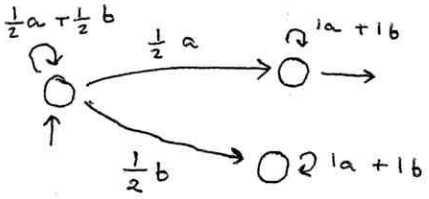
Threshold Languages

$$L_{\bowtie \alpha} = \{ w \in \Sigma^* \mid L(w) \bowtie \alpha \} \quad \text{where } \alpha \in \{ \leq, <, \geq, >, =, \neq \}$$

and L is a function realized by a ~~threshold~~ probalistic automata

Consider a slight modification of the following automaton from LECTURE - 6.

(A)



This probabilistic automaton computes the binary value denoted by a finite word in $\{a,b\}^*$ $(0.w)_2$ (fractional part) as in LECTURE-6. And by a similar argument, there are uncountably many $L_{\geq \alpha}$ for real values α. This means threshold languages are not, in general, decidable.

However, if there is a computable predicate available for $(\bowtie \alpha)$, it is trivial to decide membership in $L_{\bowtie \alpha}$ by simply evaluating the word and computing the predicate.

Since Probabilistic Automata operates over a 0-sum-free semiring, support language of an automata is regular, and membership / non-emptiness of the support language is decidable.

We will see the following claim about regularity of threshold languages.

<u>Theorem</u> Let A be a probabilistic automata. $\alpha \in [0, 1]$ is a isolated cut point if there is a $\delta > 0$ such that $\forall w \in \Sigma^*$, $|[A](w) - \alpha| > \delta$ → defn of isolated cutpoint.

$L_{>\alpha}$ is regular if α is an isolated cut point.

<u>Proof</u>

For the purpose of this proof, we will assume $\lambda^A = (1\ 0\ 0\ \ldots\ 0)$. See the discussion in LECTURE-8 to see how to convert A to this form.

Recall Myhill-Nerode Theorem: For a given language $L \subseteq \Sigma^*$, define $\equiv_L$ on Σ^* by $u \equiv_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \Leftrightarrow vw \in L$.

L is regular iff $\equiv_L$ has finite index.

A simple corollary is the following:

Suppose $\tilde{\xi} : \Sigma^* \longrightarrow S$ is such that $\tilde{\xi}(u) = \tilde{\xi}(v) \Longrightarrow u \equiv_L v$

and S has finitely many ~~states~~ elements.

Then L is regular.

<u>Proof</u> The equivalence induced by $\tilde{\xi}$ on Σ^* is a refinement of $\equiv_L$ by definition. Furthermore, this equivalence is of finite index. Thus, the equivalence $\equiv_L$ is also of finite index. $\square$

Define $\xi(u) = \lambda\mu(u) = (\mu(u)_{1,1} \;\; \mu(u)_{1,2} \;\; \cdots \;\; \mu(u)_{1,n})$

<u>Claim</u> $u \not\equiv_L v \Longrightarrow |\xi(u) - \xi(v)| \geq 4\delta$

where $|\rho| = \sum_{i=1}^{n} |\rho_i|$

We postpone the proof of the claim and analyze its consequences first.

Define $\simeq$ on Σ^* by the following:

$$u \simeq v \text{ iff } |\xi(u)_i - \xi(v)_i| \leq \frac{2\delta}{n} \text{ for each component } i$$

~~Then by the contrapositive of the claim,~~

So, $u \simeq v \Longrightarrow |\xi(u) - \xi(v)| = \sum_i |\xi(u)_i - \xi(v)_i| < n\left(\frac{2\delta}{n}\right) = 2\delta$

Thus, by the contrapositive of the claim $u \equiv_L v$.

This means that the relation $\simeq$ refines $\equiv_L$.

Here, we define $S = \dfrac{[0,1]^n}{\approx}$ where $\vec{a} \approx \vec{b}$ iff $|a_i - b_i| < \frac{2\delta}{n}$

Elements of S are $\frac{2\delta}{n}$ size cubes in $[0,1]^n$, and hence it is clear that S is finite.

We may define $\tilde{\xi}$ by the composition of ξ followed by the canonical projection to S.

With this definition of $\tilde{\xi}$, we can see the hypothesis of the claim is satisfied. Hence, the language is regular.

Now, we turn towards the proof of the claim:

Suppose w is such that $uw \in L_{\geq \alpha}$ but $vw \notin L_{\geq \alpha}$

Since α is an isolated cut point,

$$\lambda \mu(uw)\gamma \geq \alpha + \delta$$
$$\lambda \mu(vw)\gamma \leq \alpha - \delta$$

So, $\lambda(\mu(uw) - \mu(vw))\gamma \geq 2\delta$

$$\Rightarrow \lambda(\mu(u) - \mu(v))\mu(w)\gamma \geq 2\delta$$

$$\Rightarrow \sum_{i=1}^{m} (\xi(u)_i - \xi(v)_i)(\mu(w)\gamma_i) \geq 2\delta$$

But, since $\mu(w)\gamma_i$ has positive entries between 0 and 1,

$$\sum_i (\xi(u)_i - \xi(v)_i) \geq 2\delta \quad \text{—— } \circledast$$

Define

$$P := \{i \mid \xi(u)_i - \xi(v)_i > 0\}$$
$$N := \{i \mid \xi(u)_i - \xi(v)_i \leq 0\}$$

Now, we claim that $\sum_{i \in P} (\xi(u)_i - \xi(v)_i) = \sum_{i \in N} |\xi(u)_i - \xi(v)_i|$

Why? [Notice that $\sum_{i \in P} \xi(u)_i + \sum_{i \in N} \xi(u)_i = \sum_{i \in P} \xi(v)_i + \sum_{i \in N} \xi(v)_i = 1$

So, $\sum_{i \in P} \xi(u)_i - \sum_{i \in P} \xi(v)_i = \sum_{i \in N} \xi(v)_i - \sum_{i \in N} \xi(u)_i$]

$\circledast$ already gives us that $\sum_{i \in P} |\xi(u)_i - \xi(v)_i| \geq 2\delta$

By the sub-claim above,

$$\sum_{i \in N} |\xi(u)_i - \xi(v)_i| \geq 2\delta, \text{ as well}$$

So, $|\xi(u) - \xi(v)| = \sum_i |\xi(u)_i - \xi(v)_i|$

$$= \sum_{i \in P} |\xi(u)_i - \xi(v)_i| + \sum_{i \in N} |\xi(u)_i - \xi(v)_i|$$

$$\geq 2\delta + 2\delta = 4\delta.$$

That proves the claim.

QED

We will see later that given an Automaton $\mathcal{A}$ and a value α, checking if α is an isolated cut point is undecidable.

Also, note that the converse of this theorem is not true. In particular $L_{>0}$ is always regular, as we argued that the support languages of probabilistic automaton are regular. However, automaton (A) (see page 1) ~~is not~~ does not have 0 as an isolated cut point, as infinitely many rational numbers are present in any neighborhood of 0

Value Problem

The Value-α problem is the following:

Input : $\mathcal{A}$, probabilistic automaton

Qn : Is there $w \in \Sigma^*$, s.t. $[\![\mathcal{A}]\!](w) = \alpha$?

The value-0 problem is just the question whether the support language is not universal. This is decidable because the support language is regular.

Value-α problem is undecidable for any $\alpha > 0$.

We start by showing that Value-$\frac{1}{2}$ is undecidable. The reduction is from PCP problem.

PCP Problem

Input : Σ - a finite set

$f : \Sigma \to \{0,1\}^*$

$g : \Sigma \to \{0,1\}^*$

Output : Is there $w \in \Sigma^* \setminus \{\epsilon\}$ s.t. $f(w) = g(w)$?

Exercise Suppose that $f : \Sigma \to \{0,1\}^*$ is given. Construct a probabilistic automaton that computes $(f(w))_2$ on input w. Try for $f = \{a \mapsto 010 ; b \mapsto 11 ; c \mapsto 011\}$

LECTURE - 10

We continue our discussion of the value-α problem.

Input : $\mathcal{A}$

Qn : Does $\exists w \in \Sigma^*$, $[\![\mathcal{A}]\!](w) = \alpha$ hold?

This is decidable for $\alpha = 0$, as discussed before. By considering $1 - [\![\mathcal{A}]\!]$ using the discussed closure properties, we can also decide this for $\alpha = 1$.

We ~~use~~ a reduction from PCP to ~~solve~~ show undecidability of the value-$\frac{1}{2}$ problem.

First, we describe PCP:

Let Σ be a finite set

Input : $f : \Sigma \to \{0, 1\}^*$

$g : \Sigma \to \{0, 1\}^*$

Output : Does there exist $w \in \Sigma^*$ such that $\hat{f}(w) = \hat{g}(w)$?

where $\hat{f}$ and $\hat{g}$ are the lifts ~~of~~ of f and g to Σ^*

Consider the following modification of a given PCP instance:

Given $f : \Sigma \to \{0,1\}^*$, let ~~$f'(a_1 a_2 \cdots a_n) =$~~

$$f'(a) = f(a) \gg= \begin{Bmatrix} 0 \mapsto 01 \\ 1 \mapsto 11 \end{Bmatrix} \quad \text{for } a \in \Sigma$$

By way of example, if $f(a) = 010$, then $f'(a) = 011101$

Or, with more verbose notation;

if $f(a) = a_1 a_2 \ldots a_n$

then $f'(a) = a_1 1 a_2 1 \ldots a_n 1$

It is clear that this modification preserves solutions, in the sense that f and g have a solution in original PCP iff ~~they~~ f' and g' do. Thus, we may assume that $\hat{f}(w)$ ends with 1 for any $w \in \Sigma^*$.

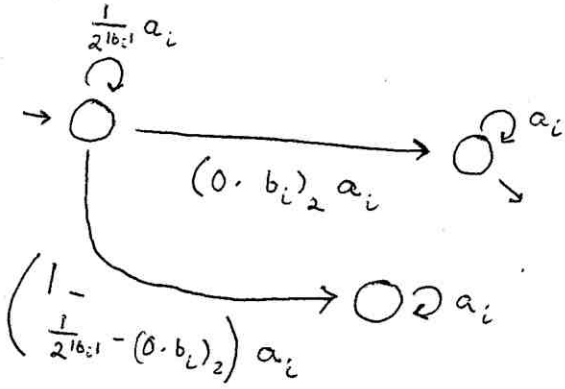
Now, we may note that $(0.\hat{f}(w))_2 = (0.\hat{g}(w))_2$ iff $\hat{f}(w) = \hat{g}(w)$

where $(\ )_2 : \Sigma^* \to [0,1]$

is the function that takes binary strings to the value they denote.

We show that given an $f : \Sigma \to \{0,1\}^*$, the function $w \mapsto (0.f(w))_2)$ is realisable by a probabilistic automata.

Suppose f takes the symbol $a_i \in \Sigma$ to the string $b_i \in \{0,1\}^*$. Then, for each such a_i, we add transitions in the following way:



We'll call this automata $\mathcal{A}_f$.

Consider the automaton that computes $\frac{1}{2}[\![\mathcal{A}_f]\!] + \frac{1}{2}(1 - [\![\mathcal{A}_g]\!])$.

We claim that this automata has a word of weight $\frac{1}{2}$ iff PCP has a solution.

Indeed, if

$$\frac{1}{2}[\![\mathcal{A}_f]\!](w) + \frac{1}{2}(1 - [\![\mathcal{A}_g]\!](w)) = \frac{1}{2}$$

$$\iff [\![\mathcal{A}_f]\!](w) = [\![\mathcal{A}_g]\!](w)$$

$$\iff (0.\hat{f}(w))_2 = (0.\hat{g}(w))_2$$

$$\iff \hat{f}(w) = \hat{g}(w)$$

Thus, we have;

<u>Proposition</u> The value-$\frac{1}{2}$ problem is undecidable.

$\blacksquare$

Next, we show that value-α is undecidable for $0 < \alpha < 1$.

<u>When $\alpha < \frac{1}{2}$</u>

Consider $\mathcal{A}$ that computes $\alpha [\![\mathcal{A}_f]\!] + \alpha (1 - [\![\mathcal{A}_g]\!])$.

By a similar argument, this automaton has a word of weight α iff $[\![\mathcal{A}_f]\!](w) = [\![\mathcal{A}_g]\!](w)$ for some w. And Hence, we can reduce PCP to the value-α problem

<u>When $\alpha > \frac{1}{2}$</u> $\mathcal{A}$ has a word of weight α iff $(1 - [\![\mathcal{A}]\!])$ has a word of weight $(1-\alpha)$.

But, $\alpha > \frac{1}{2} \Rightarrow 1 - \alpha < \frac{1}{2}$. So, this reduces to the previous case.

QED

<u>Proposition</u> Given $\mathcal{A}$, it is undecidable to check if there exists w s.t. $[\![\mathcal{A}]\!](w) \geq \alpha$

<u>Proof</u> We reduce the Value-$\frac{1}{2}$ problem to this problem.

Recall that given probabilistic machines $\mathcal{A}$ and $\mathcal{B}$, we can find C that computes $[\![\mathcal{A}]\!][\![\mathcal{B}]\!]$

We call this the <u>Hadamard Product</u>

Consider the automaton $\mathcal{B}$ that computes $([\![\mathcal{A}]\!](1-[\![\mathcal{A}]\!]))$.

$([\![\mathcal{A}]\!](1-[\![\mathcal{A}]\!]))(w) \leq \frac{1}{4}$. with equality iff $[\![\mathcal{A}]\!](w) = \frac{1}{2}$

Thus, $[\![\mathcal{A}]\!]$ has a word of weight $\frac{1}{2}$ iff $[\![\mathcal{B}]\!]$ has a word of weight $\geq \frac{1}{4}$. ∎

Next, we consider the following Problem

Input: ~~Weighted~~ Probabilistic Automata $\mathcal{A}$, Value α

Question: Is α an isolated cutpoint of $\mathcal{A}$?

We show that this is undecidable.

<u>Proof</u>

We consider the following problem:

Input: Finite Set Σ

Morphisms $f : \Sigma \to \{0,1\}^*$

$g : \Sigma \to \{0,1\}^*$

s.t. $f(a) \neq g(a)$ for some $a \in \Sigma$

Output: Is the set

$\{\text{Longest Common Prefix}(\hat{f}(w), \hat{g}(w)) \mid w \in \Sigma^*\}$

finite?

This problem is undecidable. We omit the proof.

We reduce our problem from this.

Again, let $\mathcal{A}$ be an automata that computes $\alpha[\![\mathcal{A}_f]\!] + \alpha(1-[\![\mathcal{A}_g]\!])$ as before.

If the set of longest common prefixes is infinite,

given any $\varepsilon > 0$, there is w such that $|(0.f(w))_2 - (0.g(w))_2| < \varepsilon$

and thus $|[\![\mathcal{A}]\!](w) - \alpha| < \varepsilon$

This can be witnessed by wa such that $f(a) \neq g(a)$ but $f(w)$ and $g(w)$ have a sufficiently long common prefix

On the other hand, if we have $|[\![A]\!](w) - \alpha| < \varepsilon$ for $\varepsilon < 2^{i}$, the first i bits of $f(w)$ and $g(w)$ must agree. Thus, the set of longest common prefixes is infinite iff A has α as a cluster point.

$\blacksquare$

Whether 0 or 1 is an isolated cut point is also undecidable. However, this was open for sometime.

Consider the equivalence problem for probabilistic automata

Input : A, B

Equivalence : Is $[\![A]\!] = [\![B]\!]$?

This problem is ~~[illegible]~~ decidable. Indeed, the automaton $[\![A]\!] - [\![B]\!]$, which is no longer probabilistic, can be embedded in $\mathbb{R}$. Since $\mathbb{R}$ is a field, we may use techniques from LECTURE-4 to check support non-emptiness.

LECTURE - 11

In classical automata theory, we have Kleene's Theorem: Rational Languages are ~~Regular~~ Recognizable Languages. We want an analogue for the weighted setting.

Series

Suppose $f : \Sigma^* \longrightarrow \mathbb{S}$ is a weighted language over $\mathbb{S}$, then we represent f by the formal object $\sum_{w \in \Sigma^*} f(w)\, w$, which is what we call series. Note that since Σ^* is non-commutative, the coefficient of, say, 'ab', cannot be identified with the coefficient of 'ba'.

A polynomial is a series in which all but finitely many coefficients were 0. A monomial is a polynomial with exactly one non-zero coefficient.

~~Note that~~ We shall use 1 to mean the polynomial 1, i.e, the function which has weight 1 on ε and 0 on all other words. Note that we may define scalar multiplication on the left or on the right in a natural way.

$$\lambda \sum_{w \in \Sigma^*} f(w)\, w = \sum_{w \in \Sigma^*} (\lambda \cdot f(w))\, w$$

$$\left(\sum_{w \in \Sigma^*} f(w)\, w \right) \lambda = \sum_{w \in \Sigma^*} (f(w)\, \lambda)\, w$$

Series expressions are meant to be an analogue for regular expressions in the general setting.

Metric on weighted languages and Summability

In order to define an analogue for the Kleene Star on weighted languages, we ~~define~~ need a notion of convergence for infinite sums, as we will see. Hence, we equip the space of quantitative languages with a metric.

$$d : (\Sigma^* \to \mathbb{S}) \times (\Sigma^* \to \mathbb{S}) \longrightarrow \mathbb{R}$$

$$d(f,g) = \left.\begin{array}{ll} 2^{-r} & f \neq g \\ 0 & f = g \end{array}\right\} \quad \text{where } r = \min \{ |w| \mid f(w) \neq g(w),\ w \in \Sigma^* \}$$

It is easy to check that d is a metric on $\Sigma^* \to S$. Infact, it is an ultrametric:

$$d(f,h) \leq \max(d(f,g),\ d(g,h)) \text{ for all } g$$

<u>Defn</u> Suppose $\{f_k\}_{k\in K}$ is a family of series indexed by K. $\{f_k\}_{k\in K}$ is <u>summable</u> if $\exists f$ s.t. $\forall \epsilon > 0$, $\exists I \subseteq_{finite} K$, $\forall J_{finite} \supseteq I$

$$d\left(\sum_{j\in J} f_j, f\right) < \varepsilon$$

where $\sum_{j\in J} f_j$ is an usually interpretted finite sum.

If this happens, we say $\sum_{k\in K} f_k = f$

<u>Defn</u> $\{f_k\}_{k\in K}$ is <u>locally finite</u> if for all $w \in \Sigma^*$, there are only finitely many indices $k \in K$ s.t. $f_k(w) \neq 0$

<u>Lemma</u> A locally finite family is summable. ∎

~~Lemma~~

<u>Defn</u> ~~$\{f_k\}_{k\in K}$ is prop~~ f is proper if $f(\varepsilon) = 0$

~~Defn Let f be a~~

<u>Rational Series</u>

The smallest class of weighted languages which includes all polynomials and are closed under the following operations are called rational series:

- Scalar multiplication on left and right; $(sf)(w) = s \cdot f(w)$ $\quad s \in S$
  $(fs)(w) = f(w) \cdot s$
- Binary Sum: $(f+g)(w) = f(w) + g(w)$
- Cauchy Product: $(f \cdot g)(w) = \sum_{\substack{u,v \\ uv=w}} f(u)g(v)$
- Kleene Star: Define $f^i = \begin{cases} 1 & \text{if } i = 0 \\ f \cdot f^i & i > 0 \end{cases}$

$$f^* = \sum_{i \geq 0} f^i$$

We discuss a sufficient condition for f^* to be defined.

<u>Lemma</u> (aka. f is proper) If $f(\varepsilon) = 0$, f^* is well defined.

<u>Proof</u> We show that $\{f^i\}_{i \geq 0}$ is locally finite.

Suppose $w \in \Sigma^*$ and $|w| = \ell$. Then, for any $j > \ell$, the coefficient of w in f^j would be

$$\sum_{\substack{u_1 u_2 \cdots u_j \\ u_1 u_2 \cdots u_j = w}} f(u_1) \cdot f(u_2) \cdots f(u_j).$$

But, at least one of the u_i's would be 0. So, at most ℓ indices have a non-zero value at w.

$\boxtimes$

<u>Rational Series are Recognizable</u>

- Recognizing a monomial is easy. We need an automata that matches the exact word and has the exact weight
- Binary sums are realizable via disjoint union.
- Suppose $\mathcal{A}$ and $\mathcal{B}$ are two weighted automata. Below, we construct $\mathcal{C}$ which recognizes the Cauchy Product of the corresponding functions. This construction is analogous to the corresponding construction for concatenation in usual finite state automata.

Consider the state space $Q = \mathcal{A} \sqcup \mathcal{B}$

For every transition $q_1 \xrightarrow[w \cdot a]{} q_2$ in $\mathcal{A}$,

add a corresponding transition in the new automaton.

Repeat a similar construction for transitions of $\mathcal{B}$.

For triples of states $q_f \in \mathcal{A}$ and $q_i, q \in \mathcal{B}$

s.t. $q_f \xrightarrow{w_1}$

$\xrightarrow[w_2]{} q_i \xrightarrow{w_3} q$

add the transition $q_f \xrightarrow{w_1 w_2 w_3} q$ in the new automaton

For ~~triples~~ pairs of states $q_1 \in \mathcal{A}$ and $q_2 \in \mathcal{B}$

s.t. $\xrightarrow{w_0} q_1 \xrightarrow{w_1} \xrightarrow[w_2]{} q_2$

add the initial weight $\xrightarrow{w_0 w_1 w_2} q_2$

LECTURE-12

Continuing our discussion, we proceed to show that

If f is proper and recognizable, so is f^*

Step I In step I, we make sure that there is at most one non-zero entry in $\lambda^{\mathcal{A}}$ where $\mathcal{A}$ is the automaton that we will construct. Furthermore, there are no non-zero transitions entering the initial state.

To achieve this, let $\mathcal{A} = \langle \lambda, \mu, \gamma \rangle$ be the automaton realizing f. Let Q be the set of states of $\mathcal{A}$. Add ~~an~~ a fresh state q_0 to $\mathcal{A}$ and for every $a \in \Sigma$, $q_1, q_2 \in Q$, ~~add trans~~ and transitions $\xrightarrow{w} q_1 \xrightarrow[w_1 a]{} q_2$, add the transition $q_0 \xrightarrow[w w_1 a]{} q_2$. Set the in-weights of every state except q_0 to 0. Set the in-weight of q_0 to 1.

Step II Let q_0 be the only initial state of $\mathcal{A}$. Set the out weight of q_0 to 1. Set the out-weight of all other ~~transitions~~ states to 0.

For every set of transitions in the original automaton of the form $q \xrightarrow{aw} q_f \xrightarrow[w_0]{}$ add a transition $q \xrightarrow[aw w_0]{} q_0$.

With this, we have shown that Recognizable Languages $\supseteq$ Rational Languages. We turn to prove the converse.

Recognizable Languages are Rational

In standard finite state automata theory, we construct rational expressions by eliminating states. Here, we will use a more algebraic approach.

First, we make some observations.

- $(\Sigma^{Rat}, +, \cdot, 0, 1)$ is a semiring where $+$ is series addition and $\cdot$ is Cauchy Product and 0 and 1 are the 0 and 1 polynomials.
- If S is a semiring, we may consider $M(S)$, matrices over entries in S, with an induced semiring structure, as remarked in LECTURE-1. Therefore, we will consider the Matrices over Rational Series together with their semiring operations.
- Provided that a notion of Convergence exists, we may define the $*$ operation on arbitrary semirings. $x^* := 1 + x + x^2 + \dots$. This is indeed the case with $M(\Sigma^{Rat})$

<u>Arden's Lemma for Qualitative Languages</u> If $X, L_1, L_2 \subseteq \Sigma^*$ are languages satisfying $X = L_1 X + L_2$ and $\varepsilon \notin L_1$, then $X = L_1^* L_2$

We show an analogous result for series.

<u>Proposition</u> Suppose f is a series satisfying $f = g \cdot f + h$ where g is proper. Then, $f = g^* h$

<u>Proof</u>

$$f = g \cdot f + h$$
$$= g(g \cdot f + h) + h = g^2 f + g \cdot h + h$$
$$= g^2(gf + h) + gh + h = g^3 f + g^2 h + gh + h$$

First, by plugging in $f = g^* h$, we see that $g^* h$ is a solution:

$$g(g^* h) + h$$
$$= g g^* h + h$$
$$= g(1 + g + g^2 + \dots)h + h$$
$$= (gh + g^2 h + g^3 h + \dots) + h$$
$$= h + gh + g^2 h + \dots = g^* h$$

To show uniqueness,

$$f = g \cdot f + h$$
$$= g(gf + h) + h = g^2 f + gh + h$$
$$= \dots = g^{i+1} f + (g^i + g^{i-1} + \dots + g + 1)h$$

Suppose w is a word of length i which has a non-zero coefficient in f. Then, w does not have a non-zero coefficient in $g^{i+1} f$, since g is proper. Thus, w has the same coefficient in $(g^i + g^{i-1} + \dots + g + 1)h$ as it does in $g^* h$.

We can similarly argue for any word with a non-zero coefficient in $g^* h$. Thus, all words agree on $g^* h$ and f. Thus, $g^* h$ is an unique solution.

QED

Given the automaton $\langle \lambda, \mu, \gamma \rangle$, we will interpret μ as $\mu(a) + \mu(b)$. This will give us a matrix of series. Similarly, we may think of λ and γ to be vectors of series in a natural way.

Provided we compute μ^* in the manner understood above, the entry $(\mu^*)_{ij}$ would correspond to the function computed by the automata if i was a state with in-weight 1 and j was a state with out-weight 1. Thus, the dot product $\lambda \mu^* \gamma$ would give us the rational expression we require for f.

It remains to compute the entries of μ^* as rational expressions.

Computing μ^*

First, we have that $\mu^* = \mu \cdot \mu^* + \mathbb{1}$ where $\mathbb{1}$ is the identity matrix

~~We suppose that μ~~

But, we may explicitly carry out this multiplication, and get

$$(\mu^*)_{ij} = \left(\sum_k \mu_{ik} (\mu^*)_{kj} \right) + \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

This is a set of $|Q| \times |Q|$ equations, one for each i and j.

We can eliminate each $(\mu^*)_{ij}$ one at a time, by applying Arden's Lemma successively.

LECTURE - 13

Recall from the theory of classical finite state automata theory:

If L is a language and w is a word,

$$w^{-1}L := \{v \mid wv \in L\} \text{ called the residual of } L \text{ on } w$$

Also, say that $u \equiv_L v$ iff $\forall w \in \Sigma^*$, $uw \in L \iff vw \in L$

Myhill - Nerode Theorem L is regular iff $\equiv_L$ has finite index

Or equivalently, iff $\text{Residuals}(L) = \{w^{-1}L \mid w \in \Sigma^*\}$ is finite.

We will develop a weighted counterpart of this theory.

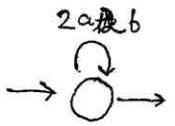
Suppose $f : \Sigma^* \to \mathbb{S}$ is a weighted language, and $w \in \Sigma^*$, then define $w^{-1}f : \Sigma^* \to \mathbb{S}$

$$x \mapsto f(wx)$$

Let $\text{Residuals}(f) = \{w^{-1}f \mid w \in \Sigma^*\}$

Is $\text{Residuals}(f)$ finite when f is recognizable? No.

For Instance, Consider $x \mapsto 2^{|x|}$, which is realised by the following automata



Notice that $w^{-1}f = x \mapsto f(wx) = x \mapsto 2^{|w|+|x|} = x \mapsto 2^{|w|}(2^{|x|})$

For every w such that $|w| \neq |v|$, $w^{-1}f \neq v^{-1}f$. Thus, there are infinitely many residuals.

To find an analogue of Myhill - Nerode, we look for a finite set of objects which generate the set of residuals.

Left $\mathbb{S}$-modules Suppose that $\mathbb{S}$ is a semiring. A (left) module over $\mathbb{S}$ is an object M endowed with a $+$-operation, which makes M a commutative monoid, along with a "scalar product" $\times : \mathbb{S} \times M \to M$ s.t.

- $s(m_1 + m_2) = sm_1 + sm_2$
- $(s_1 + s_2)m = s_1 m + s_2 m$
- $s_1(s_2 m) = (s_1 s_2)m$
- $1_{\mathbb{S}}\, m = m$
- $0_{\mathbb{S}}\, m = 0_M$
- $s\, 0_m = 0_m$

For brevity, we will call such objects modules.

A <u>submodule</u> of a module M is a subset of its elements which are closed under addition and scalar multiplication. Note that a submodule is a module in its own right.

Notice that Series, denoted $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ henceforth form a module under the addition of series and standard scalar multiplication.

A module is <u>finitely generated</u> if there is a finite set $\{m_1, \dots m_n\}$ s.t for every $m \in M$,

$$m = \sum_{i=1}^{n} \alpha_i m_i \quad \text{for some } \alpha_i \in \mathbb{S}$$

We call a submodule M of $\mathbb{S}(\langle\langle\Sigma^*\rangle\rangle)$ <u>stable</u>, if $\forall w \in \Sigma^*$, $f \in M \Rightarrow w^{-1}f \in M$

Notice that the set of polynomials are a stable submodule. Indeed, if ~~w is the non zero word~~ the length of the longest word which has a non-zero coefficient in some f is l, then for any $w \in \Sigma^* \setminus \{\varepsilon\}$, the length of the longest word with non-zero coefficient would be $l - |w|$ or smaller. So, $w^{-1}f$ is also a polynomial.

However, the submodule of polynomials ~~are~~ is not ~~stable~~ finitely generated

<u>(Analogue of Myhill Nerode Theorem for Weighted Automata)</u> Suppose $f : \Sigma^* \to \mathbb{S}$.

f is recognizable iff f is a member of a finitely generated stable submodule of $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$

<u>Proof</u> ($\Leftarrow$) Suppose $M \subseteq \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ is generated by $\{f_1 \dots f_n\}$ and $f \in M$.

Then, we can realise f by a finite state weighted automata as follows: Let the set of states be $\{f_1 \dots f_n\}$. Then, since M is stable and generated by $\{f_1 \dots f_n\}$, it is possible to pick $\{\mu^a\}_{ij}$ s.t.

$$a^{-1} f_i = \sum_{i=1}^{n} \mu^a_{ij} f_j$$

This gives us the matrices $\mu(a)$ for $a \in \Sigma$.

Now, define $\gamma_i = f_i(\varepsilon)$

and λ_j s.t $f = \sum_{j=1}^{n} \lambda_j f_j$

Then, $\langle \lambda, \mu, \gamma \rangle$ gives the automata we were looking for.

($\Longrightarrow$) Suppose f is recognizable via $\langle \lambda, \mu, \gamma \rangle$

Define f_i to be the function that takes w to $(\mu(w)\gamma)_i$

Then $f = \sum_i \lambda_i f_i$

The submodule we are looking for is the one generated by $\{f_1, f_2 \dots f_n\}$

To show that it is stable, it is enough to show that $a^{-1} f_i$ can be written as $\sum c_{ij} f_j$ for coefficients $c_{ij} \in \mathbb{S}$ for every generator f_i and every $a \in \Sigma$. Indeed, by definition, we have that

$$a^{-1} f_i = \sum_{j=1}^{n} \mu(a)_{ij} \; f_j$$

QED

<u>Aside</u> The function $a^n \mapsto 2^{2^n}$ is not recognizable over $(\mathbb{N}, +, \times)$ and $\Sigma = \{a\}$

Suppose an automaton A has $|Q|$ many states with the largest weight k. Then, the weight of a single run of an word a^n is bounded by k^n. The number of such runs is bounded by $|Q|^n$. So, $[\![A]\!](a^n) \leq (|Q|\,k)^n$. For sufficiently large n, $(k|Q|)^n < 2^{2^n}$.

<u>Comments on Hadamard Product</u>

For $f, g : \Sigma^* \to \mathbb{S}$, define $f \odot g$ by $w \mapsto f(w) \times g(w)$ where $\times$ is the semiring multiplication. If the semiring is commutative, $f \odot g$ is recognizable whenever f and g are. This follows from a construction similar to the one in LECTURE-8

<u>Exercise</u> Working on the semiring $(\Sigma^*, +, \cdot)$, suppose

$$\left.\begin{array}{l} f : a \mapsto a \\ \quad\;\; b \mapsto \varepsilon \\ g : a \mapsto a \\ \quad\;\; b \mapsto b \end{array}\right\}$$

Show that f and g are recognizable but $f \odot g$ is not.

<u>Aside</u> Weighted Automata over the semiring of languages are transducers. Several important decision problems for transducers are undecidable.