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 $\mathcal{A}' = \cancel{\text{Det} \circ \text{Rev} \circ \text{Det}}$

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

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 $\mathcal{B}$.

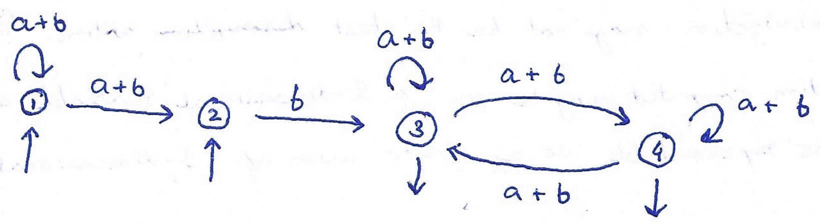
Try proving this

$\boxtimes$

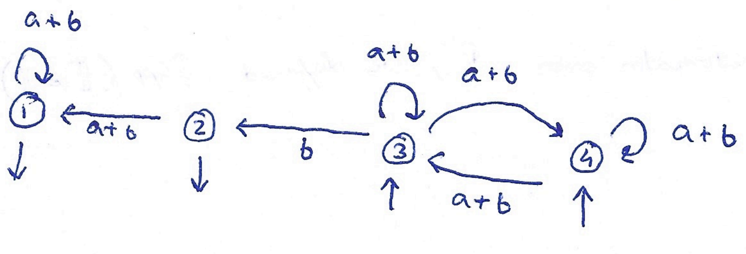
By transpose $(\mathcal{A})$, we mean the automaton $\langle \gamma, \mu^T, \lambda \rangle$ which is an analogue for the reversal operation.

Example

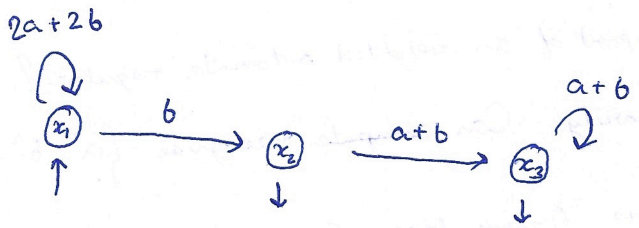
$\mathcal{A} :=$



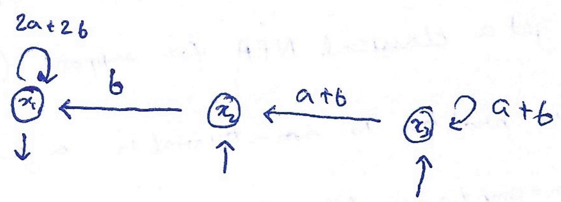
Transpose $(\mathcal{A})$ =



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



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



We reduce the last automaton further

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

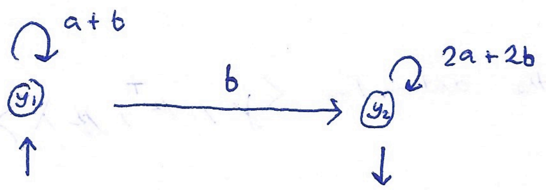
$$y_1 \circ \mu(a) = (0 \quad 1 \quad 1) = y_1$$

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

$$= y_2 + y_1$$

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

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



(Minimal Automaton)

__Aside__ 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 Supp $(\llbracket \mathcal{A} \rrbracket)$.

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, Support $(\llbracket \mathcal{A} \rrbracket)$ is regular. (Erase all weights on $\mathcal{A}$ to get a classical NFA for support $(\llbracket \mathcal{A} \rrbracket)$).

     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

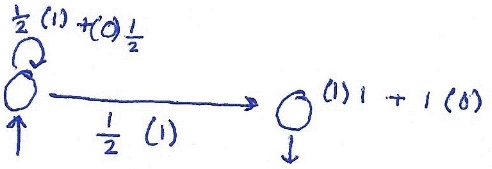
$$L_{\bowtie k} = \{ w \mid [\![A]\!](w) \bowtie k \}$$

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

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 $a_1 a_2 a_3 \dots a_n$, 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 ~~recogniz~~ 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 ~~languages~~ words via coefficients of power series.