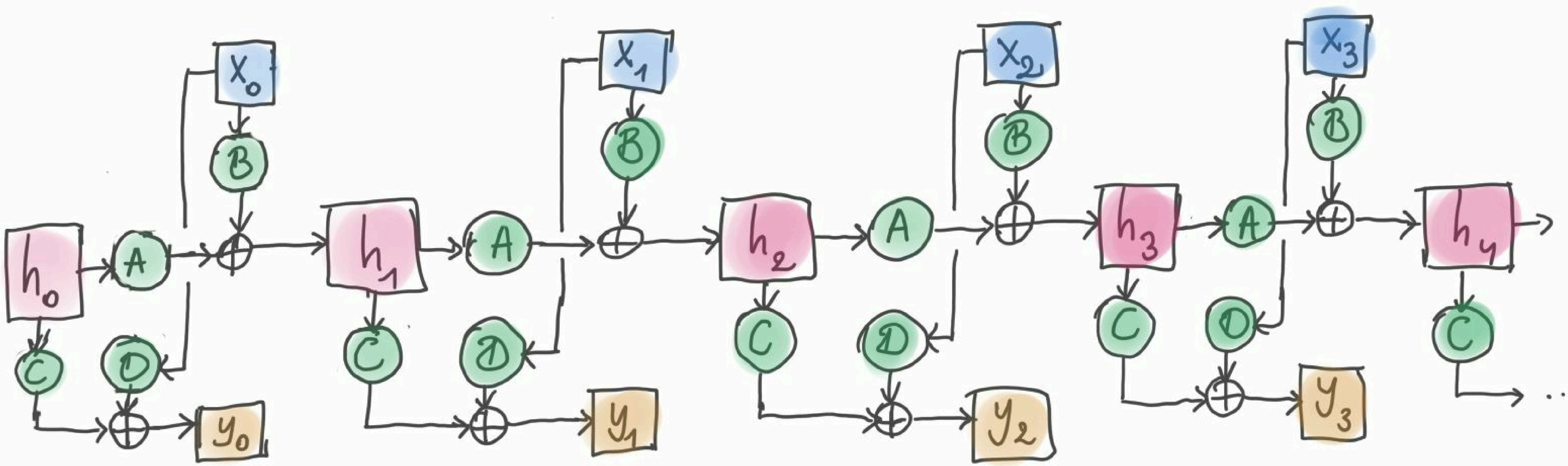
EAI Math Reading Group

State Space Models,

S4, HiPPO, Mamba

and so on, ...



Outline

1) Background on classical state-space models

2) State-Space Models in ML (S4, HiPPO)

3) Mamba

4) Surprise Galaxy brains take

5) Discussion?

Systems Theory 101

input

$$\text{state}\quad \boxed{\dot{x}(t)} = f(t, x(t), \boxed{u(t)}) \qquad u : [t_0, +\infty) \to \mathbb{R}^m$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x : [t_0, +\infty) \to \mathbb{R}^n$$

$$\text{output}\quad \boxed{y(t)} = g(t, x(t), u(t)) \qquad y : [t_0, +\infty) \to \mathbb{R}^p$$

<u>Example</u>: x: the state of the air inside a room

u: heat valve / fan speed

y: temperature measured by a thermometer

<u>Control Theory</u>: Design u to maintain desired temp.

Linear time-varying systems

$$\begin{cases} \dot{x}(t) = A(t)\,x(t) + B(t)\,u(t) \\ y(t) = C(t)\,x(t) + D(t)\,u(t) \end{cases}$$

$$x(t_0) = x_0$$

<u>General solution</u>

$$x(t) = \phi(t, t_0, x_0) = \Phi(t, t_0)\,x_0 + \int_{t_0}^{t} \Phi(t,s)\,B(s)\,u(s)\,ds$$

$$y(t) = C(t)\,\Phi(t, t_0)\,x_0 + C(t)\int_{t_0}^{t} \Phi(t,s)\,B(s)\,u(s)\,ds + D(t)\,u(t)$$

where $\Phi(t, t_0)\,x_0$ is the solution of

$$\dot{x} = A(t)\,x \quad , \quad x(t_0) = x_0$$

Linear Time-Invariant (LTI) Systems

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

$$\dot{x} = Ax$$
$$x = e^{At}x_0$$

- $\Phi(t,t_0) = e^{A(t-t_0)}$

$$\Rightarrow \quad x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-s)}Bu(s)\,ds$$
$$y(t) = Ce^{A(t-t_0)}x_0 + C\int_{t_0}^{t} e^{A(t-s)}Bu(s)\,ds + Du(t)$$

Linearity

Linear (time-varying) systems are linear transforms

- If $x_0 = 0$: linear in u ⊛
- If $u(t) = 0$: linear in x_0

⊛ This is the case for SSMs like S4, Mamba ...

$\Rightarrow$ These are <u>linear</u> transformation of sequences

Discrete time systems

$$\begin{cases} x_{t+1} = A_t x_t + B_t u_t \\ y_t = C x_t + D u_t \end{cases}$$

$$\rightsquigarrow \begin{cases} x_t = \phi_{t,t_0} x_0 + \sum_{j=t_0}^{t-1} \phi_{t,j+1} B_j u_j \\ y_t = C_t \phi_{t,t_0} x_0 + C_t \sum_{j=t_0}^{t-1} \phi_{t,j+1} B_j u_j + D_t u_t \end{cases}$$

$$\boxed{\phi_{t,t_0} = \prod_{i=t_0}^{t-1} A_i}$$

LTI case

$$\begin{cases} x_t = A^{t-t_0} x_0 + \sum_{j=t_0}^{t-1} A^{t-j-1} B u_j \\ y_t = C A^{t-t_0} x_0 + C \sum_{j=t_0}^{t-1} A^{t-j-1} B u_j + D u_t \end{cases}$$

Convolutions for LTI systems

For a discrete time LTI system, w/ $x_0 = 0$, $D = 0$

$$y_t = C \sum_{s=t_0}^{t-1} A^{t-s-1} B u_t$$

$$= \sum_{s=t_0}^{t-1} \underbrace{C A^{t-s-1} B}_{K_{t-s}} u_t$$

$$= \sum_{s=t_0}^{t-1} K_{t-s} u_t \quad \rightsquigarrow \text{discrete convolution!}$$

$$= K * u$$

$\rightsquigarrow$ Can be computed efficiently using Fourier transforms

$$K * u = \mathcal{F}^{-1}\left(\mathcal{F}(K) \cdot \mathcal{F}(u)\right)$$

1) Discretizing continuous time systems

$$\boxed{\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}} \longrightarrow \boxed{\begin{aligned} x_{t+1} &= \bar{A}x_t + \bar{B}u_t \\ y_t &= \bar{C}x_t + \bar{D}u_t \end{aligned}}$$

$$x_t = x(t\Delta) \quad y_t = y(t\Delta)$$

step size

- Zero-Order-Hold : Assuming u constant on $[t\Delta, (t+1)\Delta]$

$$x_{t+1} = \underbrace{e^{A\Delta}}_{\bar{A}} x_t + \underbrace{\left(\int_0^\Delta e^{As}\, ds\right) B}_{\bar{B}} u_t$$

$$\bar{B} = A^{-1}(\bar{A} - I)B \text{ (if } A \text{ invertible)}$$

$$y_t = \underset{\bar{C}}{\underset{\shortparallel}{C}} x_t + \underset{\bar{D}}{\underset{\shortparallel}{D}} u_t$$

① Discretization : the Bilinear transform

$$e^{A\Delta} = \sum_{k=0}^{+\infty} \frac{A^k \Delta^k}{k!} \approx I + A\Delta \qquad \text{(for } \Delta \text{ small enough)}$$

$$\rightsquigarrow \quad x_{t+1} \approx (I + A\Delta)\, x_t + \Delta B u_t \qquad \text{[Euler's method]}$$

$$e^{A\Delta} \approx (I - A\Delta)^{-1} \longrightarrow \text{[Backward Euler]}$$

$$e^{A\Delta} \approx \left(I + \tfrac{1}{2}A\Delta\right)\left(I - \tfrac{1}{2}A\Delta\right)^{-1} \qquad \text{[Bilinear transform]}$$

$$\rightarrow \boxed{\begin{aligned} \bar{A} &= \left(I - \tfrac{\Delta}{2}A\right)^{-1}\left(I + \tfrac{\Delta}{2}A\right) \\ \bar{B} &= \left(I - \tfrac{\Delta}{2}A\right)^{-1}\Delta B \end{aligned}} \qquad \bar{C} = C$$

State-Space Models for ML

- Given some sequential data $[u_0, u_1, \dots, u_L]$

  Compute some new sequence $[y_t]_{t=0}^{L}$ as

$$\begin{cases} x_{t+1} = \bar{A} x_t + \bar{B} u_t \\ y_t = \bar{C} x_t \end{cases} \quad (*)$$

  where A, B, C are learnable parameters

- Note: generally assume $(*)$ is the discretization of

$$\begin{cases} \dot{x}(t) = A x(t) + B u(t) \\ y(t) = C x(t) \end{cases}$$

$\longrightarrow$ Easier to deal with theoretically + deal w/ irregularly sampled data

A Remark on notation

Classical Systems theory	SSMs in ML
u_t : input	x_t : input sequence
x_t : state	h_t : (hidden) state
y_t : output	y_t : output sequence

We'll stick to the classical notation

HiPPO in a nutshell

The basic approach of learning A, B, C doesn't work that great

$\rightsquigarrow$ eigenvalues of A :
- $Re(\lambda) > 0$ $\rightsquigarrow$ unstable
  - $\rightarrow$ explodes
- $Re(\lambda) < 0$ $\rightsquigarrow$ stable
  - $\rightarrow$ past states vanish

$\rightarrow$ Basic idea : Fix A to keep eigenvalues with $Re\,\lambda \approx 0$

$\approx$ preserve the history of past inputs as best as possible

Orthogonal Polynomials

Given some measure μ on $\mathbb{R}$, define the inner product

$$\langle f, g \rangle_\mu = \int_{-\infty}^{+\infty} f(x)\, g(x)\, d\mu(x) = \int_{-\infty}^{+\infty} f(x)\, g(x) \underbrace{\mu(x)}_{\text{"weighting function"}} dx$$

$$\|f\|_\mu = \left(\langle f, f \rangle_\mu\right)^{\frac{1}{2}}$$

We want to construct an orthonormal set of polynomials $\{\varphi_n\}_{n \in \mathbb{N}}$ with respect to $\langle \cdot, \cdot \rangle_\mu$

$$\langle \varphi_n, \varphi_m \rangle_\mu = \delta_{nm}$$

$\rightsquigarrow$ Apply Gram-Schmidt orthogonalisation to $\{1, t, t^2, t^3, \dots\}$

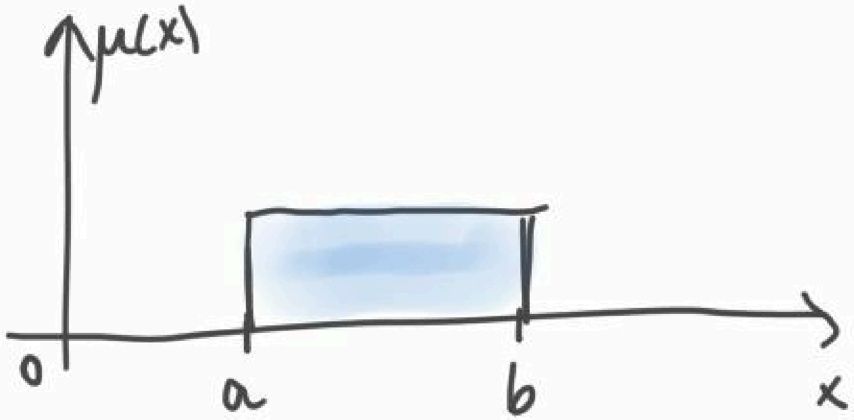
$$\varphi_0 = \frac{1}{\|1\|_\mu} \; ; \quad \varphi_n = \frac{t^n - \sum_{k=0}^{n-1} \langle t^n, \varphi_k \rangle_\mu \varphi_k}{\left\| t^n - \sum_{k=0}^{n-1} \langle t^n, \varphi_k \rangle_\mu \varphi_k \right\|_\mu}$$

Examples

$$\mu = \frac{1}{b-a}\mathbb{I}[a,b] \quad \longrightarrow \quad \text{Legendre Polynomials } [a,b]=[-1,1]$$

$$P_0 = 1, \quad P_1 = x, \quad P_2 = \frac{1}{2}(3x^2-1)$$

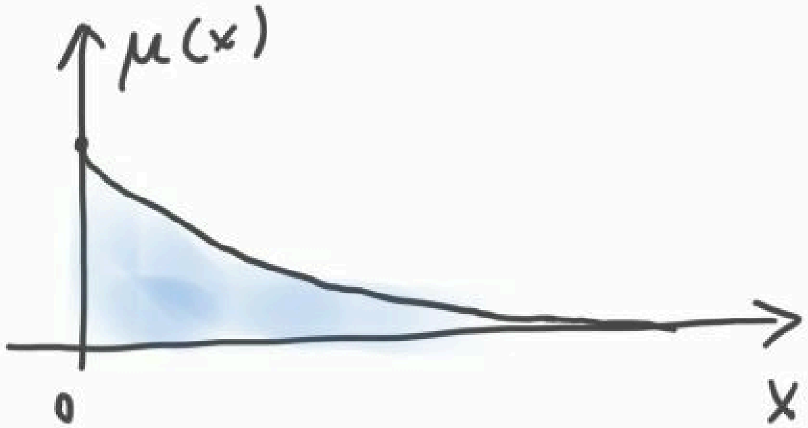
$$P_4 = \frac{1}{2}(5x^3-3x), \quad \ldots$$



$$\mu(x) = e^{-x} \quad (x \geq 0) \quad \longrightarrow \quad \text{Laguerre Polynomials}$$

$$L_0 = 1, \quad L_1 = -x+1, \quad L_2 = \frac{1}{2}(x^2-4x+2)$$

$$L_3 = \frac{1}{6}(-x^3+9x^2-18x+6), \ldots$$



$$\mu(x) = e^{-x^2} \quad \longrightarrow \quad \text{Hermite Polynomials} \ \ldots$$

Basic idea of HiPPO

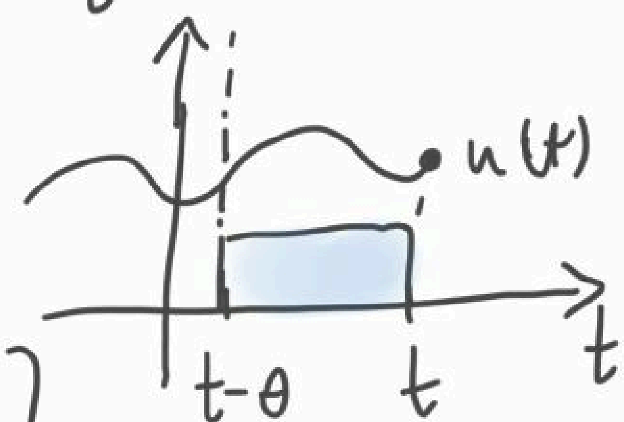
Orthogonal Polynomials can be used to approximate arbitrary* functions

$\rightarrow$ approximate $u(t)$ w/ respect to some <u>time-dependent</u> measure $\mu^{(t)}$

$$\rightsquigarrow \quad u(t) \approx \sum_{k=0}^{n} c_k(t)\, p_k^{(t)}(t)$$

The coefficients $[c_k]$ depend on time according to

$$\dot{c}(t) = \tilde{A}\, c(t) + \tilde{B}\, u(t)$$



<u>Example:</u> Legendre (LegT) $\mu^{(t)} = \frac{1}{\theta}\mathbb{I}[t-\theta, t]$

$$\rightsquigarrow \quad A_{nk} = \frac{1}{\theta}(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} \begin{cases} 1 & k \leq n \\ (-1)^{n-k} & k \geq n \end{cases} \qquad B_n = (2n+1)^{\frac{1}{2}}$$

HiPPO and S4

Use $A = \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k \\ n+1 & n = k \\ 0 & n < k \end{cases}$ triangular matrix

+ some clever parametrization to avoid numerical instability

↳ S4 : preserve long range information

- fast convolution mode for training
- sequential mode for inference

Mamba

$$\begin{cases} \dot{x}(t) = A(u(t))\,x(t) + B(u(t))\,u(t) \\ y(t) = C(u(t))\,x(t) \end{cases}$$

discretize with step $\Delta_t = \Delta(u(t))$

$$\begin{cases} x_{t+1} = \bar{A}_t\, x_t + \bar{B}_t\, u_t \\ y_t = \bar{C}_t\, x_t \end{cases}$$

(A, B, C, Δ) depend on u_t $\rightarrow$ "selection" mechanism / gating

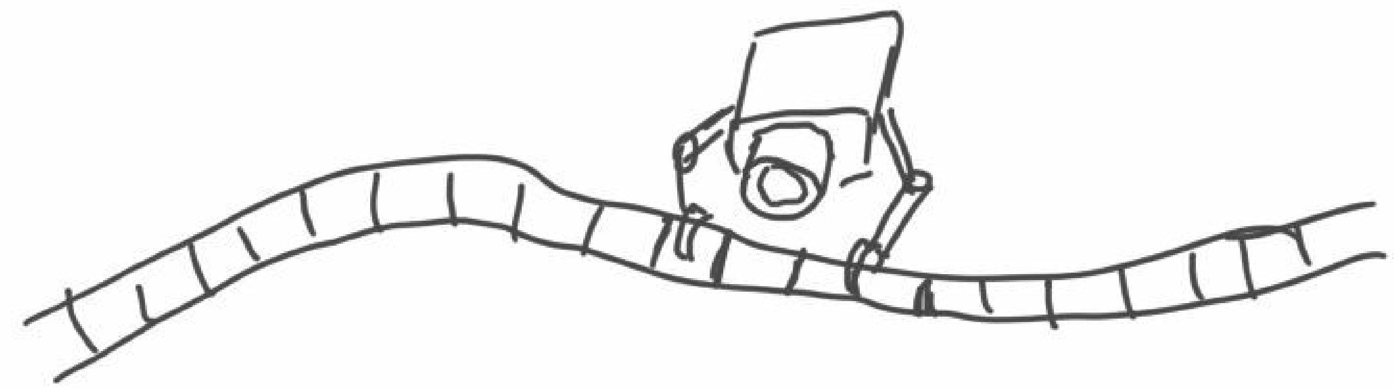
No convolutions ?

- The response y_t can't be computed as a convolution anymore
- But $\Phi_{t,t_0} = \prod_{s=t_0}^{t-1} \bar{A}_{s+1}$ matrix product $\rightsquigarrow$ <u>associative</u>

$\rightsquigarrow$ y_t can still be computed using a parallel algorithm (+ some CUDA dark magic)

Galaxy brain take: Mamba looks a bit like a Turing Machine

- Turing Machine



- infinite tape
  with symbols from Γ (finite set) $\approx$ input $[u_t]$
- machine state $\in Q$ (finite set) $\approx$ state x_t
- transition function $\delta: \underbrace{Q \times \Gamma}_{(x_t,\, u_t)} \to \underbrace{Q \times \Gamma}_{(x_{t+1},\, y_t)} \times \underbrace{\{L, R\}}_{\text{shift tape left or right}}$

From Turing Machines to "state-space model"

idea: map symbols from Q, Γ to basis vectors of $\mathbb{R}^{|Q|}, \mathbb{R}^{|\Gamma|}$

$\rightsquigarrow$ full state $$X = \underbrace{\mathbb{R}^{|Q|}}_{\substack{\text{machine}\\ \text{state}}} \oplus \underbrace{\ell^\infty(\mathbb{Z}, \mathbb{R}^{|\Gamma|})}_{\text{tape state}}$$

- Can encode δ as $$\delta_x : \underbrace{X}_{\substack{\text{current}\\ \text{state}}} \longrightarrow \underbrace{L(X)}_{\substack{\text{linear operator}\\ \text{on } X}}$$

$$(q, x) = \tilde{\delta}(\underbrace{q \otimes x_0}_{\hookrightarrow \text{ tensor product}})$$

Example

Consider the transition $(A, 0) \xrightarrow{\delta} (B, 1, R)$

- $A \to B$ can be done via a operator

$$P_A^B e_A = e_B \quad , \quad P_A^B e_C = e_C \quad \forall C \neq A$$

- same for $0 \to 1 \quad \rightsquigarrow D_0^1$
- shifting the tape can be done via the shift-operator

$$S^1 (\ldots, x_{-1}, x_0, x_1, \ldots) \longmapsto (\ldots, x_{-2}, x_{-1}, x_0, \ldots)$$

$$\rightsquigarrow \quad \delta_X(q, x) = P_A^B \oplus (S^1 \circ D_0^1)$$

$$\rightsquigarrow \left[\delta_X(q, x)\right](q, X) = \left(P_A^B q, S^1 D_0^1 x\right)$$

Linear Systems are Turing Machines that can only go left (and are bilinear)

$$\begin{cases} x_{t+1} = \bar{A}(x_t, u_t)\, x_t + \bar{B}(x_t, u_t)\, u_t \\ y_t = \bar{C}(x_t, u_t)\, x_t \end{cases}$$

Mamba is a Turing Machine that only goes left and whose transitions only depend on the tape

$$x_{t+1} = \bar{A}(u_t)\, x_t + \bar{B}(u_t)\, u_t$$

$$y_t = \bar{C}(u_t)\, x_t$$