# Representation of FSMs in RNNs

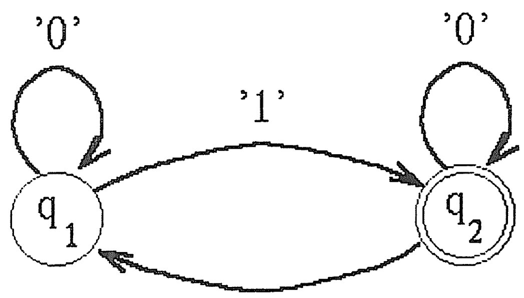
- A *Mealy machine* $\mathcal{M} = (I, O, S, q_0, \delta, \eta)$ is a six-tuple, where
  - $I$ is an alphabet of $m$ input symbols,
  - $O$ is an alphabet of $p$ output symbols,
  - $S$ is a set of $n$ states,
  - $q_0$ is a start state,
  - $\delta : I \times S \Rightarrow S$ is a state transition function,
  - $\eta : I \times S \Rightarrow O$ is an output function.
- In a *Moore machine* $\mathcal{M} = (I, O, S, q_0, \delta, \eta)$, the output function only depends on the states, $\eta : S \Rightarrow O$.

- *DFAs* and *DUFAs* are particular cases of *Moore machines*.

# Representation of transition function

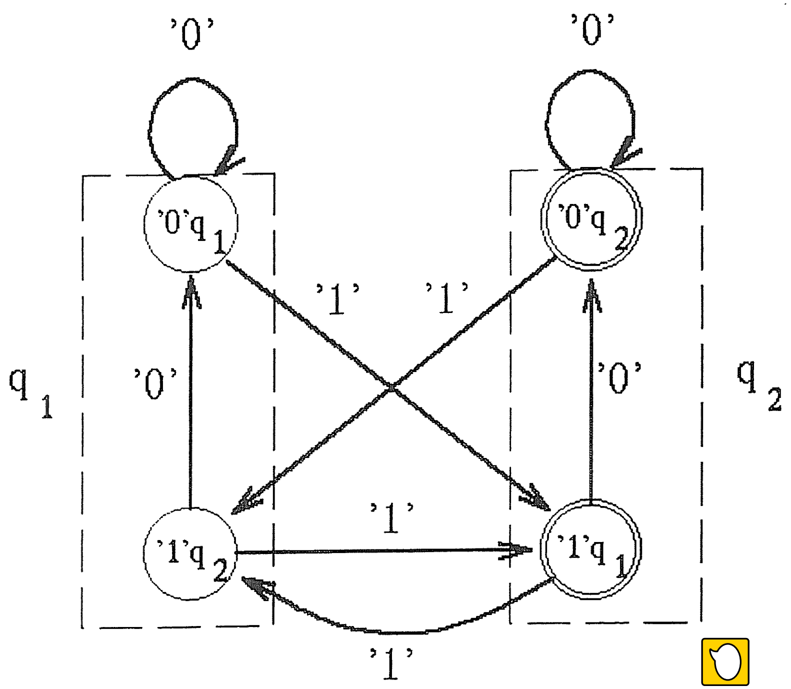
- Given an *input encoding* and a *state encoding*, the *state transition function* $\delta$ can be represented as a *linear system* $A_\delta W_\delta = B_\delta$, where
    - $A_\delta$ *(D x E)* is a matrix of neuron inputs,
    - $W_\delta$ *(E x N)* is a (transposed) matrix of neuron weights,
    - $B_\delta$ *(D x N)* is a matrix of neuron net-inputs,
    - $D = mn$ is the number of transitions of the FSM $\mathcal{M}$, and
    - $E$ is the number of weights of each neuron.
- In a *first-order SLRNN* $\mathcal{N}$, $rank(A_\delta) <= m+n-1$

    $\Rightarrow$ *first-order SLRNNs cannot implement all* $\delta$ *'s*
- In a *second-order SLRNN* $\mathcal{N}$ with *local encoding (N=n)*, $rank(A_\delta)=mn$ $\Rightarrow$ *second-order SLRNNs can implement all* $\delta$ *'s*




$$\begin{array}{c} \delta('0', q_1) \\ \delta('0', q_2) \\ \delta('1', q_1) \\ \delta('1', q_2) \end{array} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} W = \begin{pmatrix} H & -H \\ -H & H \\ -H & H \\ H & -H \end{pmatrix} \begin{array}{c} q_1 \\ q_2 \\ q_2 \\ q_1 \end{array}$$

First-order SLRNN system




$$\begin{array}{c} \delta('0', q_1) \\ \delta('0', q_2) \\ \delta('1', q_1) \\ \delta('1', q_2) \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} W = \begin{pmatrix} H & --H \\ -H & H \\ -H & H \\ H & --H \end{pmatrix} \begin{array}{c} q_1 \\ q_2 \\ q_2 \\ q_1 \end{array}$$

Second-order SLRNN system of odd-parity recognizer

$$\begin{array}{c} \delta('0','0'q_1) \\ \delta('0','1'q_2) \\ \delta('0','0'q_2) \\ \delta('0','1'q_1) \\ \delta('1','0'q_1) \\ \delta('1','1'q_2) \\ \delta('1','0'q_2) \\ \delta('1','1'q_1) \end{array} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} W = \begin{pmatrix} 2\omega+\theta & \omega+\theta & \omega+\theta & \theta \\ 2\omega+\theta & \omega+\theta & \omega+\theta & \theta \\ \omega+\theta & 2\omega+\theta & \theta & \omega+\theta \\ \omega+\theta & 2\omega+\theta & \theta & \omega+\theta \\ \omega+\theta & \theta & 2\omega+\theta & \omega+\theta \\ \omega+\theta & \theta & 2\omega+\theta & \omega+\theta \\ \theta & \omega+\theta & \omega+\theta & 2\omega+\theta \\ \theta & \omega+\theta & \omega+\theta & 2\omega+\theta \end{pmatrix} \begin{array}{c} '0'q_1 \\ '0'q_1 \\ '0'q_2 \\ '0'q_2 \\ '1'q_1 \\ '1'q_1 \\ '1'q_2 \\ '1'q_2 \end{array}$$

First-order SLRNN system of maximally split odd-parity recognizer

# Representation of output function

- Given also an *output encoding*, the *output function* $\eta$ can be represented in
  - a *second-order SLRNN* as a *linear system* $A_\eta W_\eta = B_\eta$ where
    - $A_\eta = A_\delta$,
    - $W_\eta$ is a submatrix of $W_\delta$ (with only *P* columns)
    - $B_\eta$ is a submatrix of $B_\delta$ (with only *P* columns)
  - the output layer of a *first-order 2L-ASLRNN* as an additional *linear system* $A_\eta W_\eta = B_\eta$ where
    - $A_\eta$ $(D \times (N+1))$ is a matrix of output unit inputs,
    - $W_\eta$ $((N+1) \times P)$ is a matrix of output unit weights, and
    - $B_\eta$ $(D \times P)$ is a matrix of output unit net-inputs.

- Both systems can be solved using a *local encoding*.

# Representation of FSMs in RNNs - Summary

- A *second-order SLRNN* or *2L-ASLRNN* with $M=m$, $N=n$, and $P=p$ can represent any Mealy machine.
- A *first-order 2L-ASLRNN* with $M=m$, $N=mn$, and $P=p$ can represent an equivalent FSM (with state split) to any Mealy machine.
- *DFAs*, *DUFAs*, and *stochastic DFAs* can be *inserted* in the above RNN architectures with *different activation functions* through *linear system solving*.
- The *inserted symbolic rules* can be *preserved* during subsequent training by means of a *constrained learning algorithm*.

# FSA (and UFSA) Insertion into 2L-ASLRNNs

① *Establish underdetermined linear systems* $A_\delta W_\delta = B_\delta$ and $A_\eta W_\eta = B_\eta$, that represent the $\delta$ and $\eta$ functions of the *inserted FSA*, using a *2L-ASLRNN* with more hidden units than required to solve the systems $(N > n)$.

② *Initialize the weights of the hidden and output units* to any of the solutions $W_\delta$ and $W_\eta$, *respectively*, that result from solving the two underdetermined systems.

# Constrained neural learning method

- Adjust *independent weights* by gradient-descent and update the *dependent weights* to keep linear relations given by system solution (*search in a linear subspace of weights*).
- *Independent weights* are changed according to

$$\Delta w_{kl}(t) = -\alpha \left( \frac{\partial E(t)}{\partial w_{kl}} + \sum_{w_{ka} \in D(W_k)} \frac{\partial E(t)}{\partial w_{ka}} \frac{\partial w_{ka}}{\partial w_{kl}} \right)$$

where $D(W_k)$ is the subset of *dependent weights* of unit $k$

and $\frac{\partial w_{ka}}{\partial w_{kl}}$ are known constants.