

# Basic Properties of Attribute Grammar Encoding

---

Talib S. Hussain and Roger A. Browse  
Queen's University, Kingston, Ontario

# Main Points

---

- ◆ Attribute grammars generate classes of neural network architectures
- ◆ Attribute grammar derivation trees are a useful genetic encoding of neural networks
- ◆ Attribute grammar productions facilitate the design of typed genetic operators
- ◆ Interesting variations on the selection of adaptation points may be explored

# Attribute Grammar Production

---

*Production*

*Context-Free Production*

*Name*

SEQ:  $M_1 \rightarrow M_2 M_3$

*(synthesized)*

In\_Nodes **of**  $M_1 :=$  In\_Nodes **of**  $M_2$ ;

Out\_Nodes **of**  $M_1 :=$  Out\_Nodes **of**  $M_3$ ;

All\_Nodes **of**  $M_1 :=$  [All\_Nodes **of**  $M_2 \cup$  All\_Nodes **of**  $M_3$ ];

Connections **of**  $M_1 :=$  [Connections **of**  $M_2 \cup$  Connections **of**  $M_3$

$\cup$  *full\_connect*(Out\_Nodes **of**  $M_2$ , In\_Nodes **of**  $M_3$ )];

*Attribute*

*Evaluation*

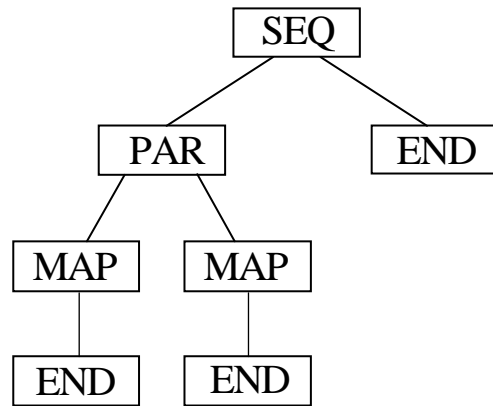
*Functions*

# Context-Free Portion of Attribute Grammar for Basic Neural Networks

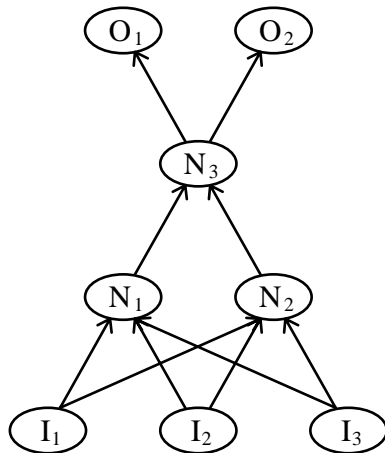
---

Start:         $S \rightarrow M$   
SEQ:          $M_1 \rightarrow M_2 M_3$   
PAR:          $M \rightarrow L_1 L_2$   
MAP:          $L \rightarrow M$   
END:          $M \rightarrow \text{node}$

# Network Generation Process



*Derivation Tree*

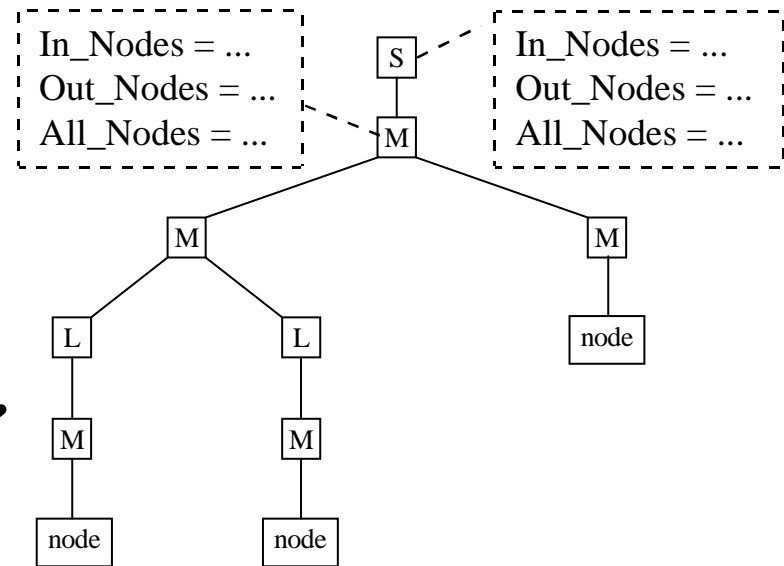


*Neural Network Structure*

→ Evaluate Attributes

↙ Extract Attributes

→ Interpret Structure

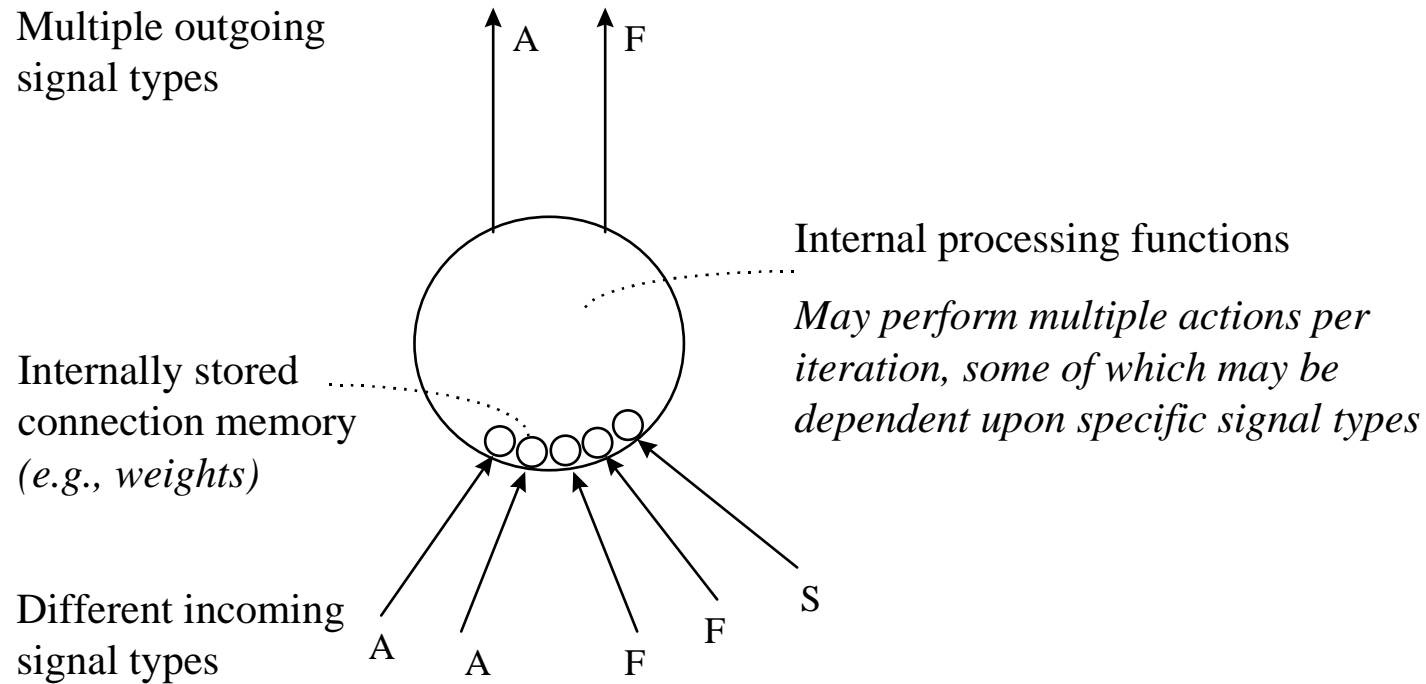


*Attributed Parse Tree*

*Executable Neural Network*

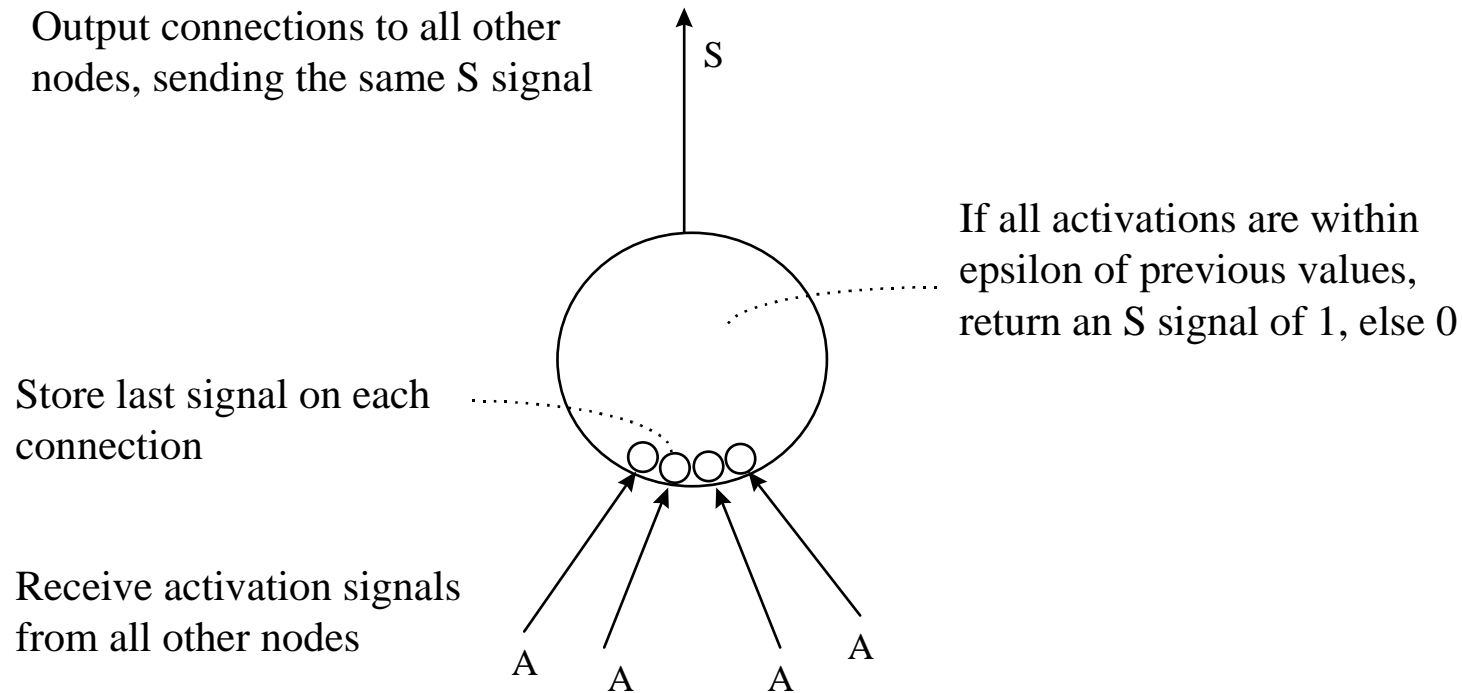
# Basic Neuron Model

---



# Control Node: Stable Activation Detector

---



# Illustrative 'Control' Productions

---

Start:        S    → N

Main:        N    → C M

Control:    C    → a b c

a:            stable activation detector

b:            stable feedback detector

c:            learning complete detector

# Attribute Grammar Derivation Trees as Genetic Encoding

---

- ◆ Subtrees exhibit strong semantic identity
- ◆ Trees are compact compared to phenotype
- ◆ Grammar design promotes modularity of the phenotype (i.e., modular topology)
- ◆ New productions may enhance compactness
- ◆ Coarse phenotype structure may be represented near root of tree and fine structure near leaves of tree

# Genotypic Compactness

---

CLONE1:  $M \rightarrow C$

*(inherited)*

Number\_Clones of C := 5;

*(synthesized)*

In\_Nodes of M := In\_Nodes of C;

Out\_Nodes of M := Out\_Nodes of C;

All\_Nodes of M := All\_Nodes of C;

Connections of M := Connections of C;

CLONE2:  $C \rightarrow M$

*(synthesized)*

In\_Nodes of C := In\_Nodes of M  $\times$  Number\_Clones of C;

Out\_Nodes of C := Out\_Nodes of M  $\times$  Number\_Clones of C;

All\_Nodes of C := All\_Nodes of M  $\times$  Number\_Clones of C;

Connections of C := Connections of M  $\times$  Number\_Clones of C;

# “Strongly-Typed” Design

---

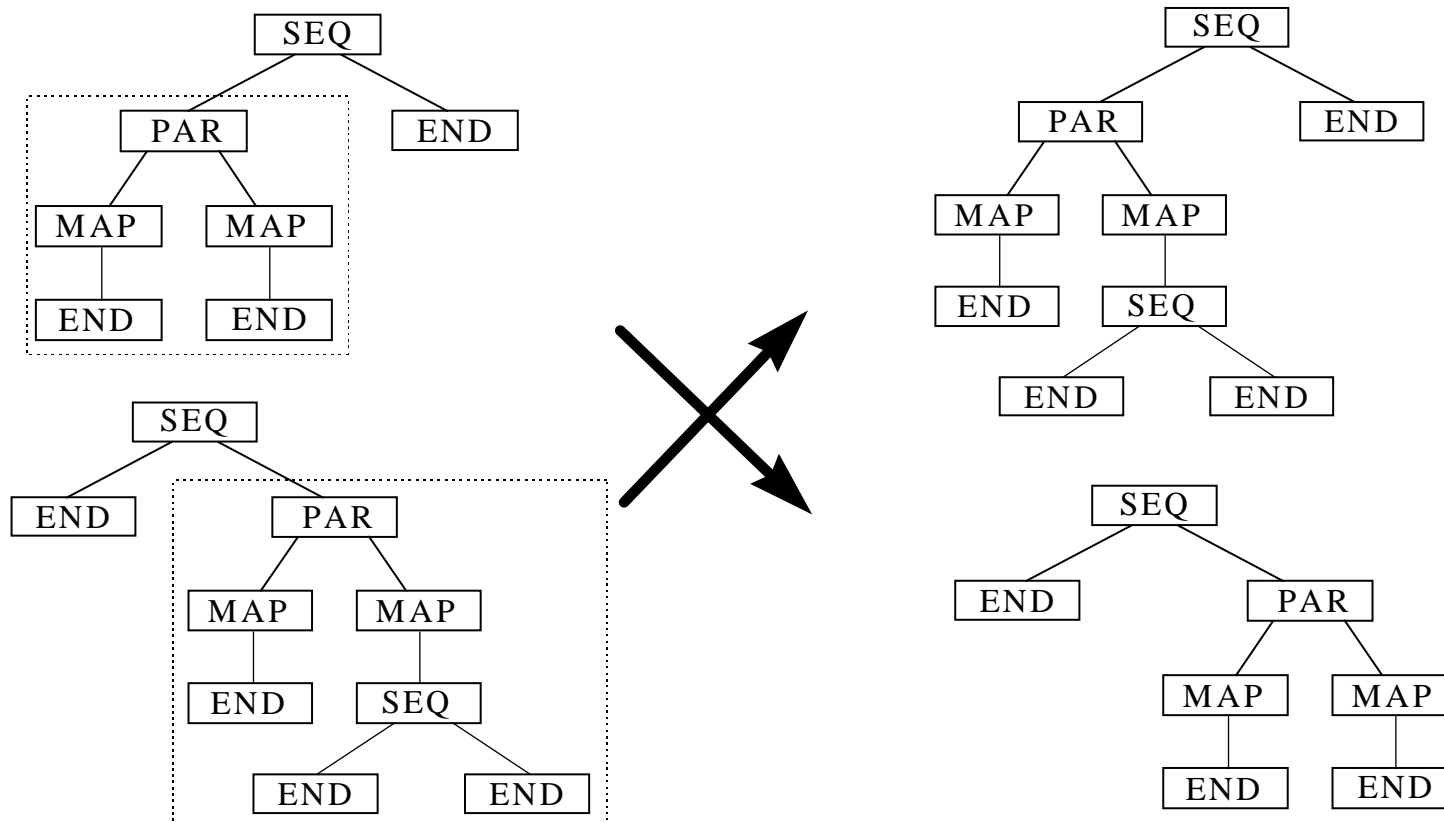
- ◆ Each non-terminal symbol in the grammar may be considered as a distinct ‘type’
- ◆ Genetic operators may be designed to preserve symbols, thereby preserving type
- ◆ Ensures legality of complex genotypes
- ◆ Allows logical manipulations of phenotype
- ◆ Permits invariance of certain phenotypic characteristics

# Constrained Crossover Operators

---

## Subtree Crossover1:

Select a node in each tree, each with same production rule, and swap the subtrees rooted by those nodes.

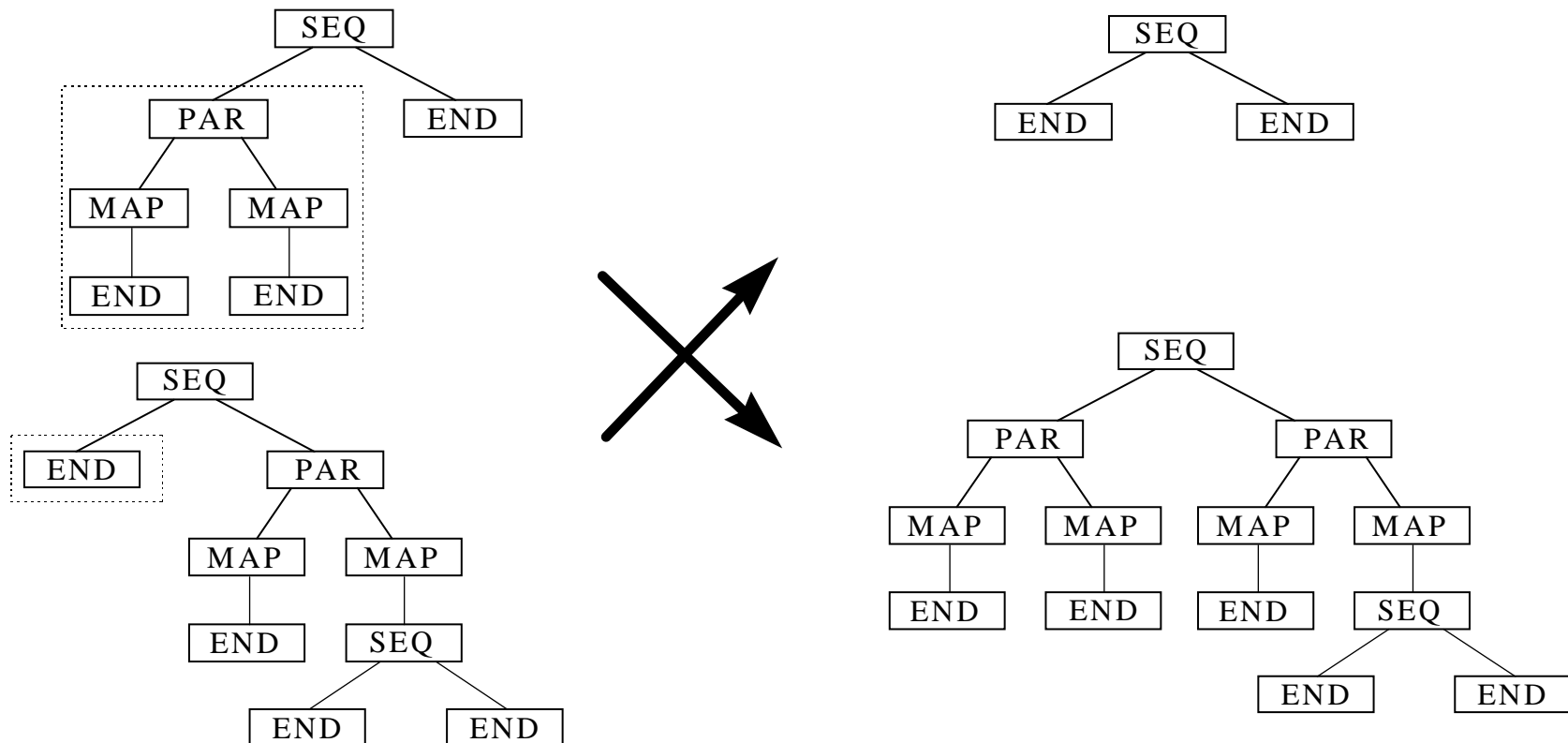


# Constrained Crossover Operators

---

## Subtree Crossover2:

Select a node in each tree, each having the same left-hand non-terminal, and swap the subtrees rooted by those nodes.

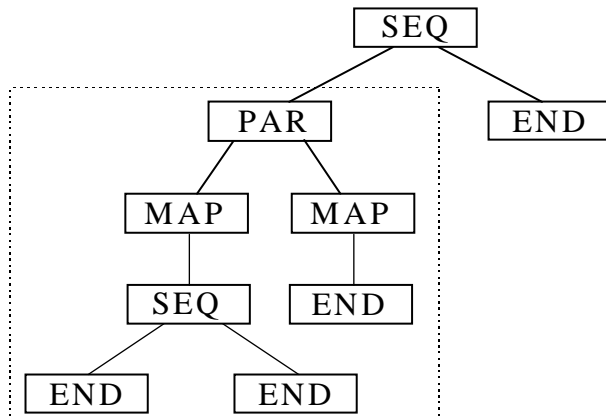


# Constrained Mutation Operators

---

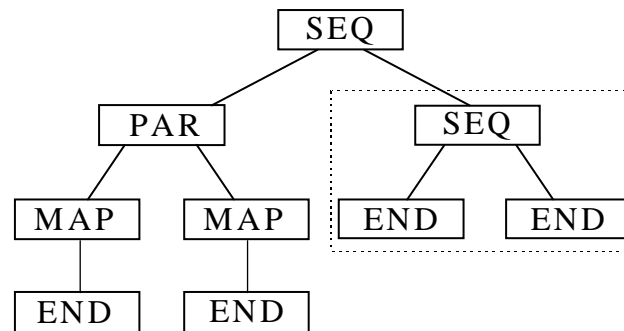
## Mutation1:

Select a node and use grammar to generate a new subtree rooted by the same production rule.



## Mutation2:

Select a node and use grammar to generate a new subtree rooted by a production rule that has the same left-hand symbol.



# Reinforcement learning with probabilistic adaptation points

---

- ① Assign each non-terminal in the grammar a probability value
- ② In selecting an adaptation point, first select what non-terminal symbol will be chosen
- ③ Then, select randomly from all internal nodes in the genetic tree which contain that symbol
- ④ If offspring are improved, increase the probability associated with the selected non-terminal symbol

# Why probabilistic adaptation points?

---

- ◆ Adds new dimension to strongly typed approaches to evolutionary computation
- ◆ Different symbols may correspond to different granularities of NN structures
- ◆ Consequences of genetic manipulations may be manipulated dynamically
- ◆ May be compared with spatially-based adaptation point selection heuristics

# Contributions

---

- ◆ Formal NN specification framework
- ◆ New approach to NN structural regularities
- ◆ Systematic exploration of NN classes
- ◆ New methods of constraining tree-based evolutionary algorithms

# Current and Future Research

---

- ◆ Representation of existing architectures
- ◆ Typed genetic operators for AG encodings
- ◆ Probabilistic selection of adaptation points
- ◆ Application of attribute grammar encoding to other complex phenotypes