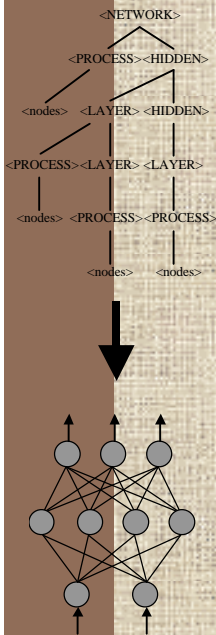


# Evolving Neural Networks using Attribute Grammars

Talib S. Hussain and Roger A. Browse

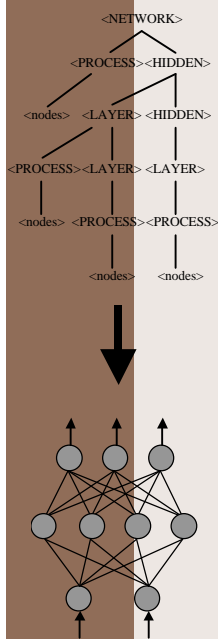
Queen's University

Kingston, Ontario, Canada



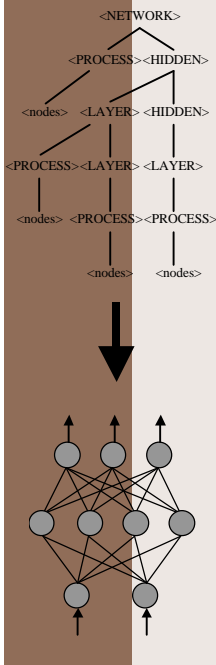
# Main Points

- There is a need for good representations of complex neural networks
- Attribute grammars exhibit good representational power and are useful for genetic codes
- Attribute grammars generate classes of neural network architectures
- Hierarchical grammars permit useful genetic representation of complex neural networks
- Assignment of probability values to the grammar productions and grammar non-terminals, and use of those probabilities in our genetic operations permits explicit biasing of genetic search.



# Attribute Grammars

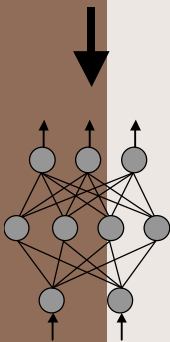
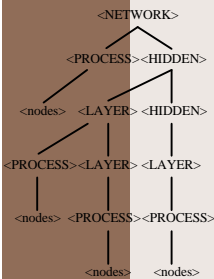
- Extension of context free grammars
  1. Assignment of attribute values to the non-terminal and terminal symbols of the grammar
  2. Addition of context-sensitive attribute evaluation functions to the grammar productions.
- Still produces nice simple parse trees
- Hierarchical design permits distinction between functionally different components.
- Attributes permit the explicit inclusion of complex structural and functional constraints on the neural networks generated.
- A complete network representation may be formed in the attributes of the root symbol



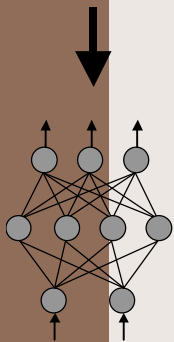
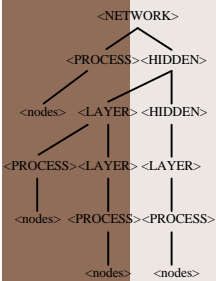
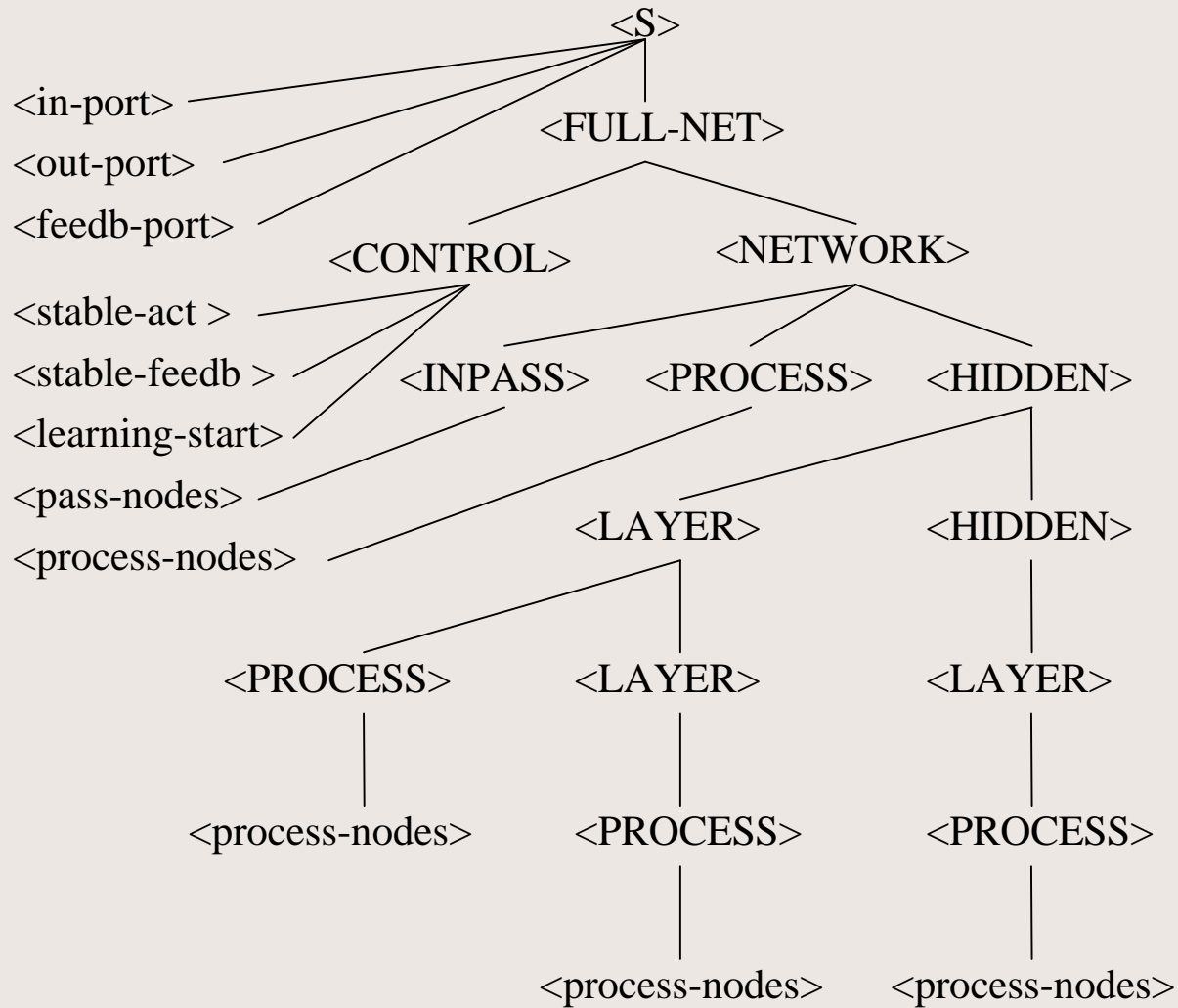
# Example: Context-Free Base

- For example, here is the context-free portion of an attribute grammar for back-propagation networks:

<S>	→ <in-port> <out-port> <feedb-port> <FULL-NET>
<FULL-NET>	→ <CONTROL> <NETWORK>
<CONTROL>	→ <stable-act> <stable-feedb> <learning-start>
<NETWORK>	→ <INPASS> <PROCESS> <HIDDEN>
<HIDDEN>	→ <LAYER> <HIDDEN>
<LAYER>	→ <LAYER>
<LAYER>	→ <PROCESS> <LAYER>
<PROCESS>	→ <PROCESS>
<PROCESS>	→ <process-nodes>
<INPASS>	→ <pass-nodes>



# Example: Parse Tree



# Example: Inherited Attributes

- Through inherited attributes, we may imposed top-down constraints, such as limiting the number of layers that are generated, and limited the size of each layer.

*Attribute  
Evaluation  
Functions*

**<HIDDEN><sub>1</sub> → <LAYER> <HIDDEN><sub>2</sub>**  
(*inherited*)

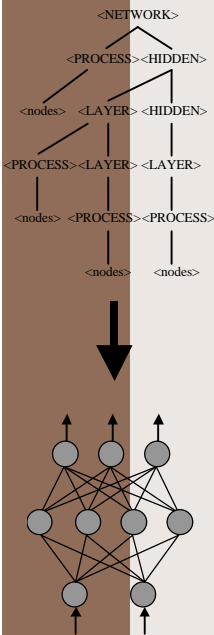
**<LAYER>.max\_size := <HIDDEN><sub>1</sub>.max\_size**

**<HIDDEN><sub>2</sub>.max\_layers := *max*((<HIDDEN><sub>1</sub>.max\_layers - 1), 0)**

**<HIDDEN><sub>2</sub>.max\_size := *if* <HIDDEN><sub>2</sub>.max\_layers > 0  
*then* <HIDDEN><sub>1</sub>.max\_size  
*else* 0**

**<HIDDEN> → <LAYER>**  
(*inherited*)

**<LAYER>.max\_size := <HIDDEN>.max\_size**



# Example: Inherited Attributes

- Through synthesized attributes, we may collect information from other symbols in the production and pass that information up to the next level of the parse tree.
- Eventually, we can collect a full description of the neural network at the root symbol.

$\langle \text{HIDDEN} \rangle_1 \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle_2$

(*synthesized*)

$\langle \text{HIDDEN} \rangle_1.\text{in\_nodes} := \langle \text{LAYER} \rangle.\text{all\_nodes}$

$\langle \text{HIDDEN} \rangle_1.\text{out\_nodes} := \text{if non-empty}(\langle \text{HIDDEN} \rangle_2.\text{out\_nodes})$

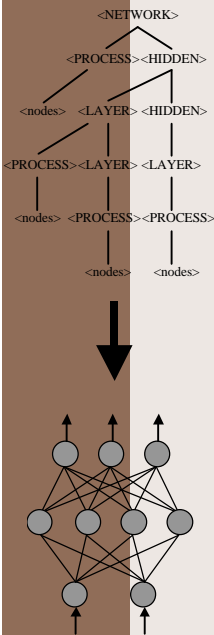
*then*  $\langle \text{HIDDEN} \rangle_2.\text{out\_nodes}$

*else*  $\langle \text{LAYER} \rangle.\text{all\_nodes}$

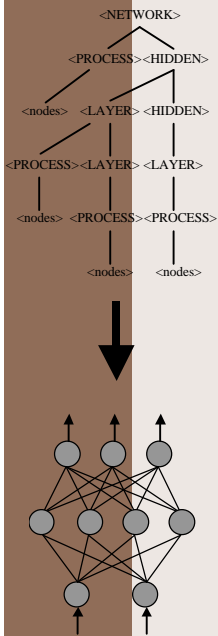
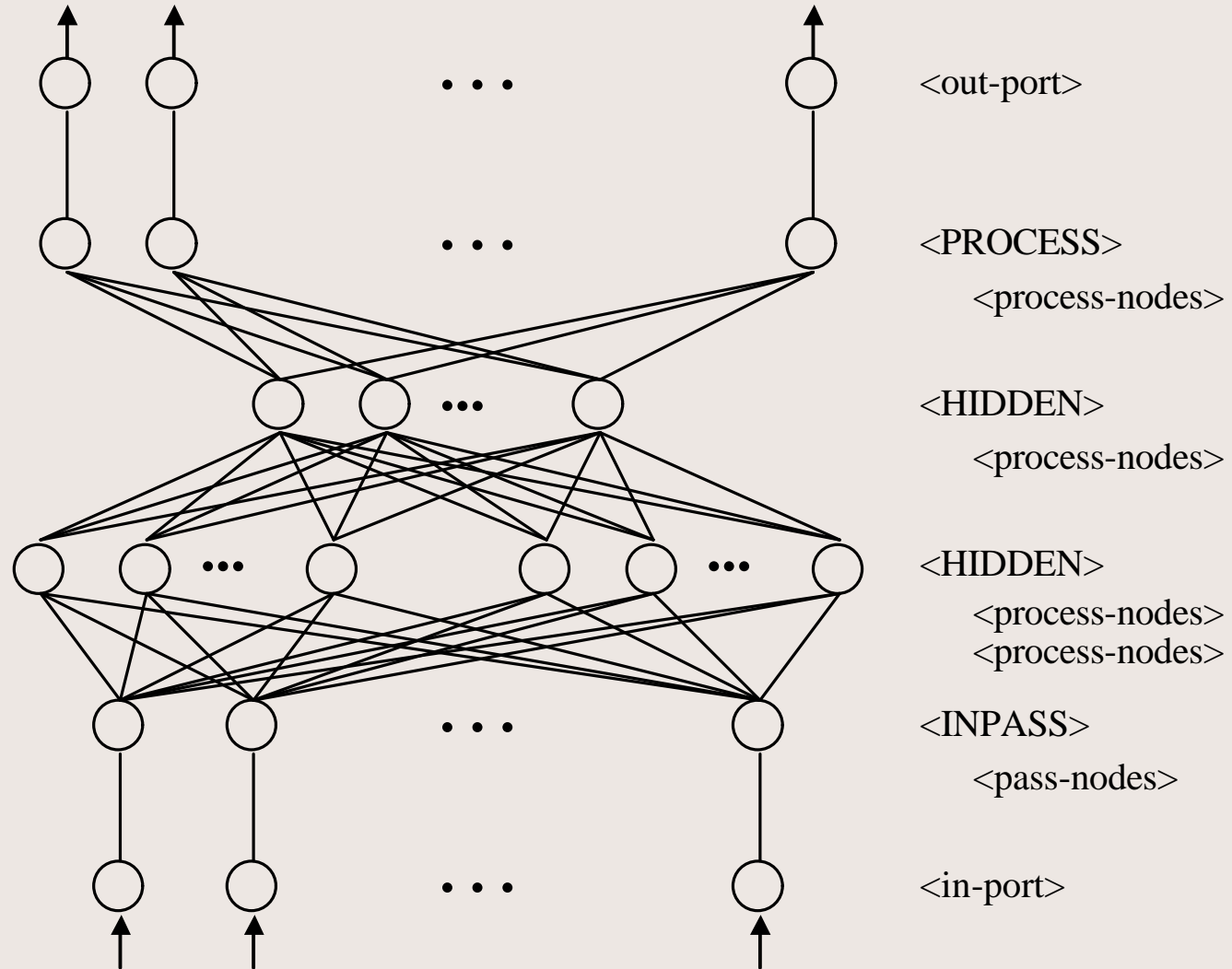
$\langle \text{HIDDEN} \rangle_1.\text{all\_nodes} := \langle \text{LAYER} \rangle.\text{all\_nodes} \cup \langle \text{HIDDEN} \rangle_2.\text{all\_nodes}$

$\langle \text{HIDDEN} \rangle_1.\text{connections} := \langle \text{HIDDEN} \rangle_2.\text{connections} \cup$

*fully\_connect*( $\langle \text{LAYER} \rangle.\text{all\_nodes}, \langle \text{HIDDEN} \rangle_2.\text{in\_nodes}$ )

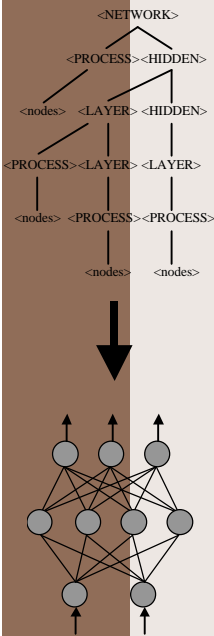


# Example: Network Depicted by Parse Tree



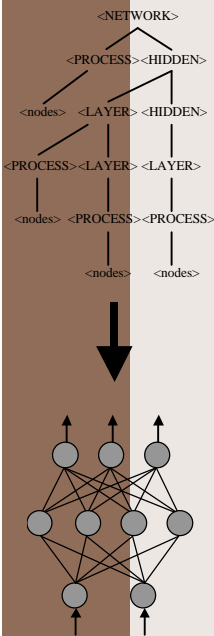
# Attribute Grammar Derivation Trees as Genetic Encoding

- 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



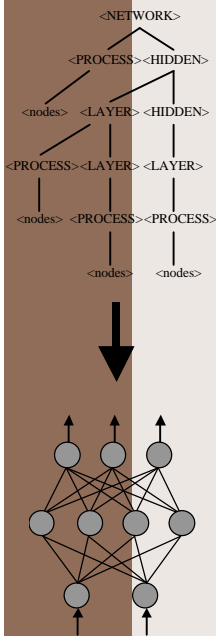
# “Strongly-Typed” Design

- 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



# Production Probabilities

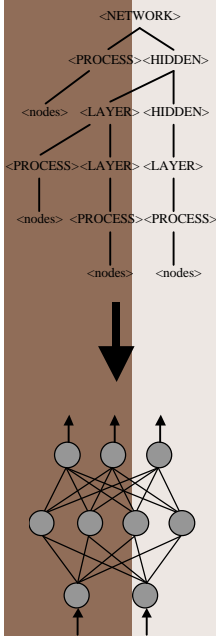
- The production rules  $\mathbf{R}$  of the grammar may be grouped according to the left-hand non-terminal symbol  $s$  that they expand.
  - The resulting sets of productions  $\mathbf{R}_s$  may contain one or more members.
- Within each set, probability values  $p_i$  may be assigned to the members  $r_i \in \mathbf{R}_s$ 
  - For sets with only one member, the probability assigned to that production is 1.



# Production Probabilities

- For example, we could assign probabilities to our earlier grammar as follows:

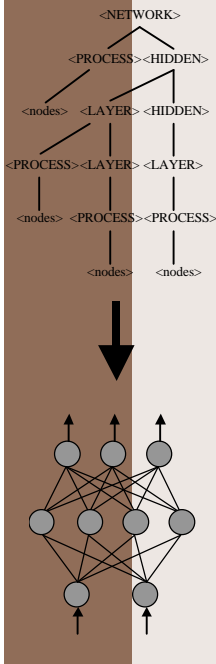
$$\begin{aligned}
 \mathbf{R}_{\langle S \rangle} &: \{ [ p_1 = 1, \langle S \rangle \rightarrow \langle \text{in-port} \rangle \langle \text{out-port} \rangle \langle \text{feedb-port} \rangle \langle \text{FULL-NET} \rangle ] \} \\
 \mathbf{R}_{\langle \text{FULL-NET} \rangle} &: \{ [ p_1 = 1, \langle \text{FULL-NET} \rangle \rightarrow \langle \text{CONTROL} \rangle \langle \text{NETWORK} \rangle ] \} \\
 \mathbf{R}_{\langle \text{CONTROL} \rangle} &: \{ [ p_1 = 1, \langle \text{CONTROL} \rangle \rightarrow \langle \text{stable-act} \rangle \langle \text{stable-feedb} \rangle \langle \text{learning-start} \rangle ] \} \\
 \mathbf{R}_{\langle \text{NETWORK} \rangle} &: \{ [ p_1 = 1, \langle \text{NETWORK} \rangle \rightarrow \langle \text{INPASS} \rangle \langle \text{PROCESS} \rangle \langle \text{HIDDEN} \rangle ] \} \\
 \mathbf{R}_{\langle \text{HIDDEN} \rangle} &: \{ [ p_1 = 0.7, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle ] \\
 &\quad [ p_2 = 0.3, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle ] \} \\
 \mathbf{R}_{\langle \text{LAYER} \rangle} &: \{ [ p_1 = 0.4, \langle \text{LAYER} \rangle \rightarrow \langle \text{PROCESS} \rangle \langle \text{LAYER} \rangle ] \\
 &\quad [ p_2 = 0.6, \langle \text{LAYER} \rangle \rightarrow \langle \text{PROCESS} \rangle ] \} \\
 \mathbf{R}_{\langle \text{PROCESS} \rangle} &: \{ [ p_1 = 1, \langle \text{PROCESS} \rangle \rightarrow \langle \text{process-nodes} \rangle ] \} \\
 \mathbf{R}_{\langle \text{INPASS} \rangle} &: \{ [ p_1 = 1, \langle \text{INPASS} \rangle \rightarrow \langle \text{pass-nodes} \rangle ] \}
 \end{aligned}$$



- During the random generation of a parse tree, production probabilities may be used to select which production rule should be used to expand a given node in the tree.

# Production Probabilities: Why?

- In a “normal”, “un-biased” grammar, each rule that expands a given symbol is usually regarded as having an equal chance of being selected.
  - e.g.,  $R_{\langle \text{HIDDEN} \rangle} : \{ [p_1 = 0.5, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle ] [p_2 = 0.5, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle ] \}$
  - Every time  $\langle \text{HIDDEN} \rangle$  is expanded, we are just as likely to stop adding layers as we are to continue adding layers.



- While this is random with respect to the use of productions, it **does not** provide a uniform random distribution of the network structures.
  - e.g., a network structure with  $n$  layers (i.e., the sentence  $\langle \text{LAYER} \rangle^n$ ) will have a  $0.5^n$  chance of being generated.
- Thus, the “normal” approach is actually biased (e.g., towards networks with few layers). It is unlikely that the needs of the evolutionary algorithm will be served by such a bias in all applications.

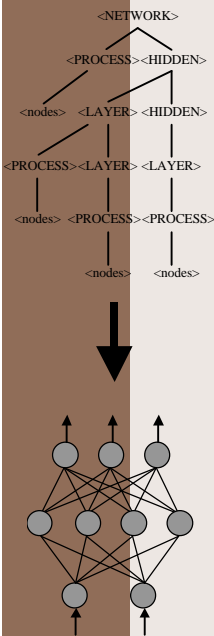
# Production Probabilities and Biased Parse Tree Creation

- Through the use of production probabilities, we may explicitly impose a specific bias upon the structures that are generated.

- e.g.,  $R_{\langle \text{HIDDEN} \rangle} : \{ [p_1 = X, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle ] [p_2 = 1-X, \langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle ] \}$
- A network structure with n layers will have a more general  $X^{n-1} (1-X)$  chance of being generated.

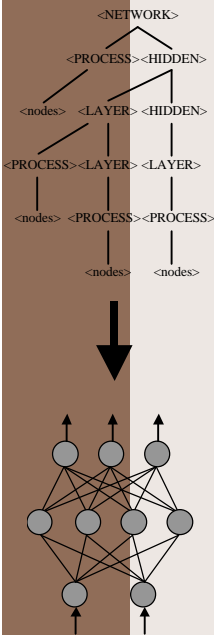
- We may thus:

- Bias the initial population in an evolutionary algorithm
- Bias the mutations that occur during reproduction



# Symbol Probabilities and Biased Genetic Operators

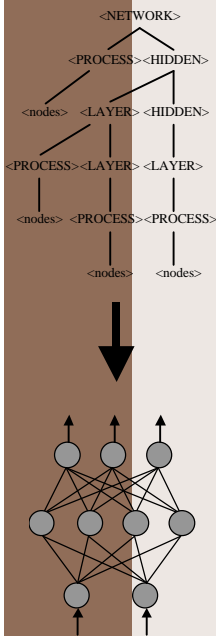
- Each non-terminal symbol  $s$  in the grammar may be assigned a probability value  $r_s$
- These values can be used by typed genetic operators when determining the point(s) of manipulation in the parent trees.
- Given a parse tree, or gene  $g$ , the set  $S_g$  of all non-terminal symbols contained within  $g$  may be determined.



- To perform subtree crossover between two trees, an internal node in each tree must be chosen (selection point).
- To preserve type, both selection points must be nodes with the same non-terminal grammar symbol

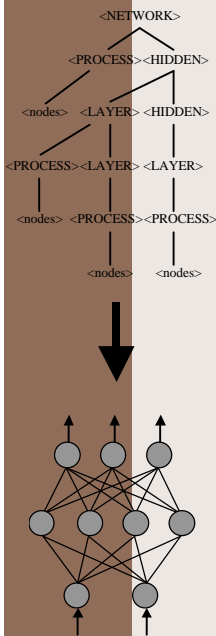
# Biased Subtree Crossover

- Biased subtree crossover on genes  $\mathbf{g}_1$  and  $\mathbf{g}_2$ :
  1. Find  $\mathbf{V} = \mathbf{S}_1 \cap \mathbf{S}_2$
  2. Each member  $\nu \in \mathbf{V}$  has associated with it a “likelihood”  $l_\nu$  of selection (where  $l_\nu = r_\nu$ )
  3. Normalize likelihoods  $l_\nu$  of all  $\nu \in \mathbf{V}$
  4. Probabilistically select  $\nu_r \in \mathbf{V}$  as the “reproduction symbol”
  5. Identify all nodes in  $\mathbf{g}_1$  that match  $\nu_r$  and randomly select one.
  6. Identify all nodes in  $\mathbf{g}_2$  that match  $\nu_r$  and randomly select one.
  7. Swap the subtrees rooted at the selected nodes.



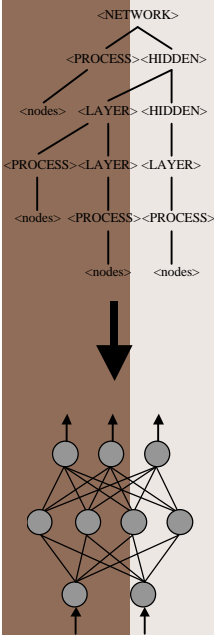
# Biased Subtree Mutation

- Biased subtree mutation on gene  $\mathbf{g}$ :
  1. Each member  $s_i \in \mathbf{S}_{\mathbf{g}}$  has associated with it a “likelihood”  $l_i$  of selection (where  $l_i = r_{s_i}$ )
  2. Normalize likelihoods  $l_i$
  3. Probabilistically select  $s_r \in \mathbf{S}_{\mathbf{g}}$  as the “reproduction symbol”
  4. Identify all nodes in  $\mathbf{g}$  that match  $s_r$  and randomly select one.
  5. Replace the subtree rooted at that node with a randomly created subtree rooted with the same symbol  $s_r$ 
    - If production probabilities are also used, then weight the random decisions made during the creation of the new subtree as appropriate.



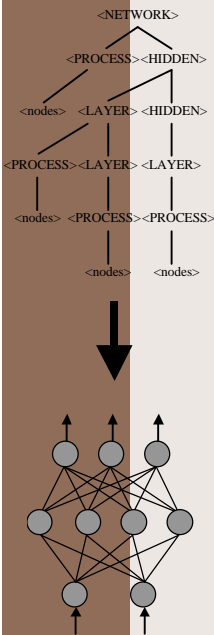
# Symbol Probabilities: Why?

- In a typical approach to crossover and mutation, the points of reproduction in the tree are chosen completely at random.
- This has two consequences.
  1. The point of reproduction are more likely to be near the bottom of the tree. In the worst case, if the tree is full, then there is a 50% chance that changes will be made at the leaves.
  2. If the tree has more nodes containing one symbol (e.g. 20 A's and 5 B's), we are more likely to crossover at a node containing the latter symbol (e.g., A).



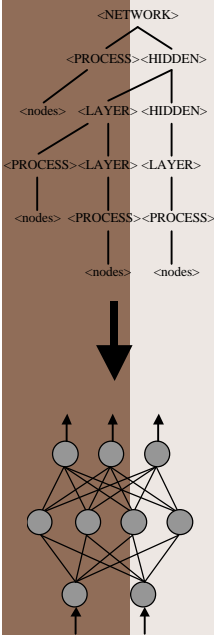
# Symbol Probabilities: Why?

- In an untyped, traditional genetic programming algorithm, these two consequences are not highly relevant since every symbol is equally likely to be generated, and may occur at any location in the tree.
- In a strongly typed genetic programming, however, both of these might highly bias the search to exploring changes primarily to a small set of the grammar symbols.
- This is especially true when the grammar is hierarchical in nature.



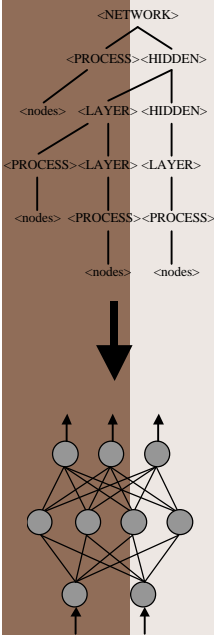
# Symbol Probabilities: Why?

- For example, in our sample grammar, a given parse tree that contains  $X$  `<HIDDEN>` symbols will always contain at least  $X$  `<LAYER>` symbols, and quite likely many more.
- Further, the `<LAYER>` symbols will be closer to the bottom of the tree than the `<HIDDEN>` symbols.
- Thus, a straightforward random selection of points of reproduction will result in the evolutionary algorithm always being biased towards exploring changes at the `<LAYER>` symbols (i.e., changes in the size of layers rather than the number of layers).



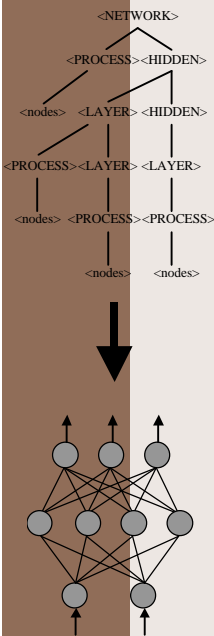
# Symbol Probabilities: Why?

- If we select based on symbols rather than nodes in the tree, we can explicitly impose a specific bias upon the structures that are generated.
- For example, in our grammar, the symbols correspond to different aspects of the neural network structure, some macroscopic in scope and some microscopic.
  - The traditional random selection will bias towards microscopic changes. This may cause the search to converge to a poor region of the search space too early.
  - Through symbol probability-based genetic operators, we can weight the search to explore more macroscopic changes if desired.
- We thus design our operators based upon the meaning of the symbols themselves rather than their prevalence in the individual genes.



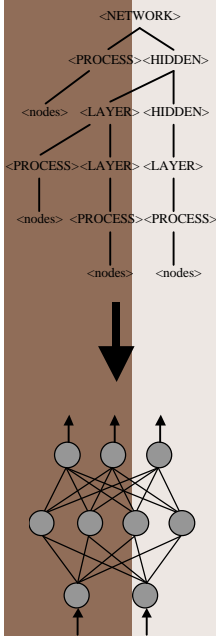
# Dynamic Biases

- A given set of production probabilities and symbol probabilities will determine a given bias in the genetic search.
- It may be the case that at different times during the genetic search, one type of bias may be more beneficial than another.
- We can allow for dynamic biases by including applying reinforcement learning to the two sets of probabilities.

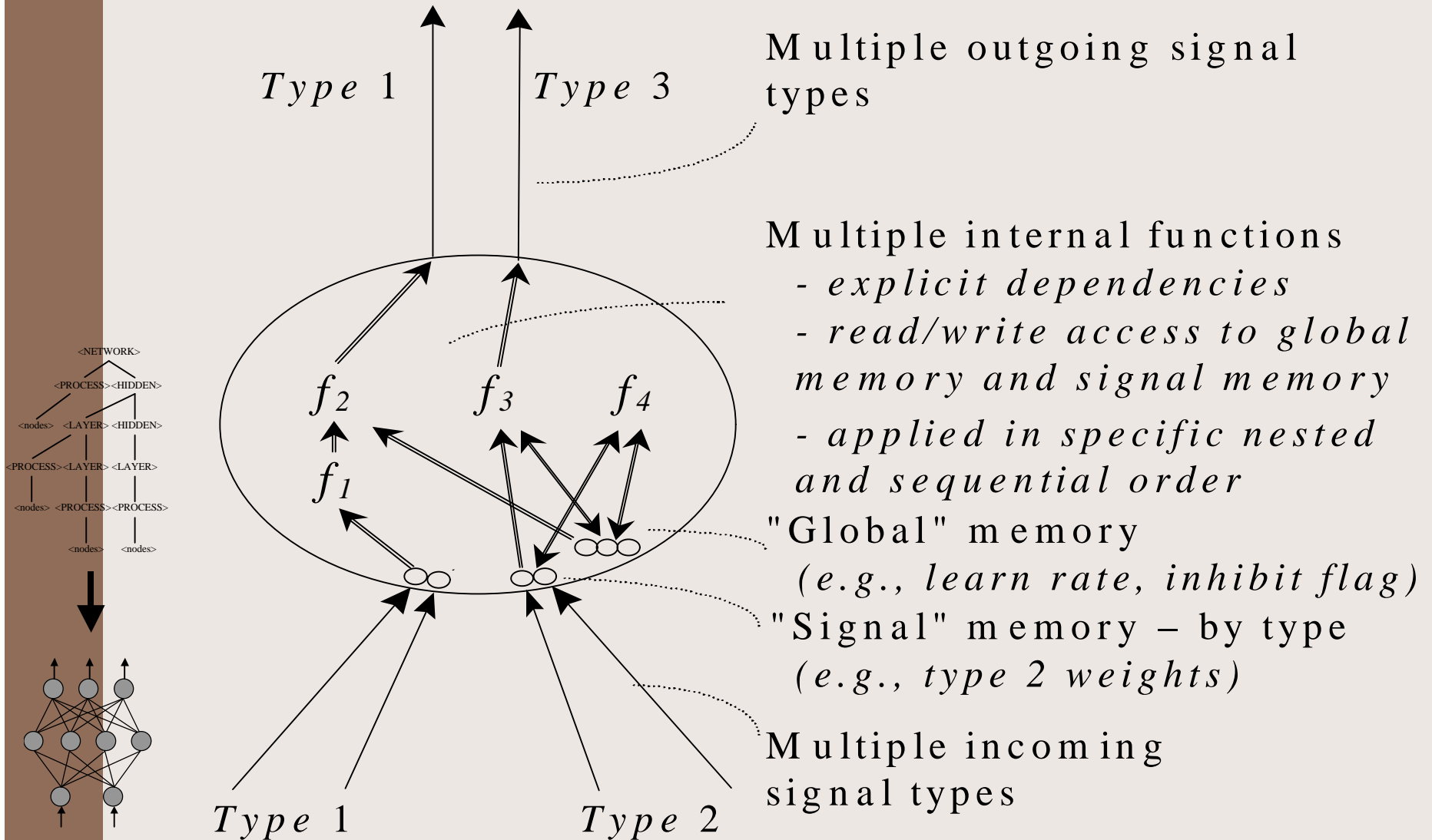


# Flexibility of Representation

- One of the main benefits of using an attribute grammar representation as a genetic encoding of neural networks is their flexibility.
- Our sample grammar illustrates the representation of only a single model, and only of a limited number of network properties.
- We can extend our grammar in many ways, as well as design attribute grammars for many other network models.
- One extension is the explicit specification of neuron functionality.

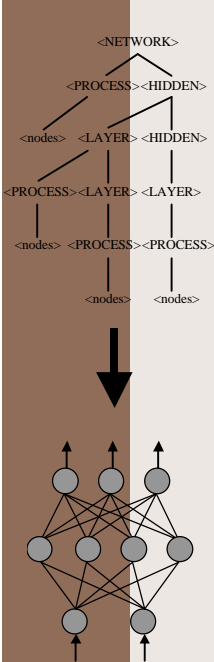


# Enhanced Neuron Model



# Node Specification: Context Free Base

- A node is regarded as consisting of a sequence of function applications and a set of memory variables. Function may take other functions or variables as parameters:



<NODE>

→ <F-SEQUENCE>

<F-SEQUENCE>

→ <FUNCTION> <F-SEQUENCE>

→ <FUNCTION>

<FUNCTION>

→ <f1> <PARAMETER> <PARAMETER>

→ <f2> <PARAMETER>

→ <f3> <PARAMETER> <PARAMETER>

<PARAMETER>

→ <FUNCTION>

→ <VARIABLE>

<VARIABLE>

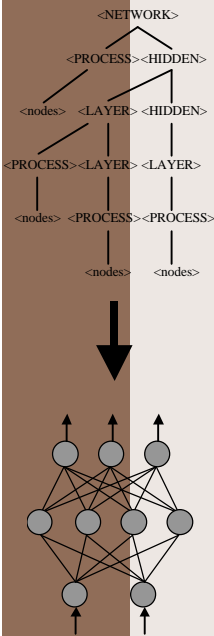
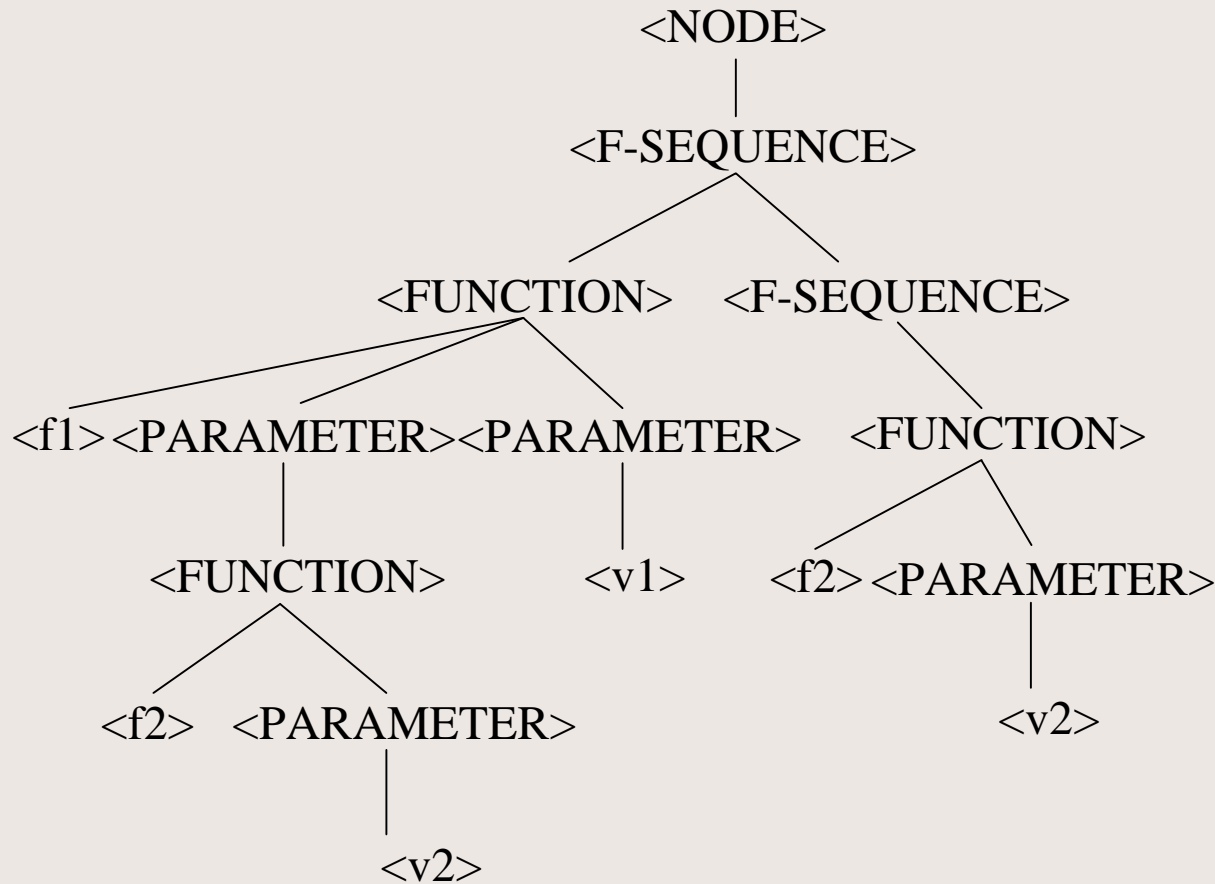
→ <v1>

→ <v2>



# Sample Parse Tree

- The following is a sample subtree specifying a single node



# Conclusions

- Novel representation for neural networks
- Conducive to genetic manipulation
- Use of a new typed genetic design that
  - Explicitly specifies the selection biases of genetic operators
  - Permits those biases to change over time.
- May be extended to include specification of neuron functionality
  - In other publications: extended to specify details of the actual learning algorithm and extended to specify other neuron models

