In Proceedings of the *2nd Government Neural Network Workshop*, 1991.

ASOCS: Towards Bridging Neural Network and Artificial Intelligence Learning

Tony Martinez

Computer Science Department, Brigham Young University
801-378-6464, E-mail:martinez@cs.byu.edu

Presented at *2nd Government Neural Network Workshop, 1991*

Introduction

A new class of connectionist architectures is presented called ASOCS (Adaptive Self-Organizing Concurrent Systems) [3,4]. ASOCS models support efficient computation through self-organized learning and parallel execution. Learning is done through the incremental presentation of rules and/or examples. Data types include Boolean and multi-state variables; recent models support analog variables. The model incorporates rules into an adaptive logic network in a parallel and self organizing fashion. The system itself resolves inconsistencies and generalizes as the rules are presented. After an introduction to the ASOCS paradigm, the abstract introduces current research thrusts which significantly increase the power and applicability of ASOCS models. For simplicity, we discuss only boolean mappings in the ASOCS overview.

ASOCS Overview

In processing mode, ASOCS supports fully parallel execution on actual inputs according to the learned rules. The adaptive logic network acts as a parallel hardware circuit during execution, mapping *n* input boolean vectors into *m* output boolean vectors, in a combinatoric fashion.

The overall philosophy of ASOCS follows the high level goals of current neural network models. However, the mechanisms of learning and execution vary significantly. The ASOCS logic network is topologically dynamic with the network growing to efficiently fit the specific application. Current ASOCS models are based on digital nodes. ASOCS also supports use of symbolic and heuristic learning mechanisms, thus combining the parallelism and distributed nature of connectionist computing, with the potential power of AI symbolic learning. A proof of concept ASOCS chip has been developed [1].

ASOCS models have two significant advantages over other learning models;

a) they are guaranteed to learn any arbitrary set of legal rules.

b) learning time is both fast and bounded (linear with the depth of the network, and logarithmic with the number of nodes).

Application targets for ASOCS include adaptive logic devices, logical inference, robotics, embedded systems, and pattern recognition.

A number of formal ASOCS models have been proposed to date [5,6,7]. In this brief paper we present a model known as Priority ASOCS (PASOCS) [2] as an example of a concrete ASOCS system. We choose PASOCS due to its simplicity and ease of presentation in this short format. PASOCS is a localist ASOCS, while other ASOCS models use a distributed paradigm and a dynamic multilayered topology [5,7].

A PASOCS model has the following attributes. Priority ASOCS:

- processes inputs in the form of Boolean variables and outputs Boolean results;
- accepts rules made up of a conjunction of boolean inputs which imply a boolean output;
- learns new rules over time;
- automatically resolves rule conflicts;
- combines specific rules into more general rules where appropriate;
- maintains an associated priority with each rule.

We begin with the black box of figure 1, having four input variables {a,b,c,d} and the single output variable z. In general, an ASOCS maps *n* input variables into *m* output variables where the size of *m* and *n* do not need to be known a priori. The system allocates input and output lines as new variables are introduced in rules.

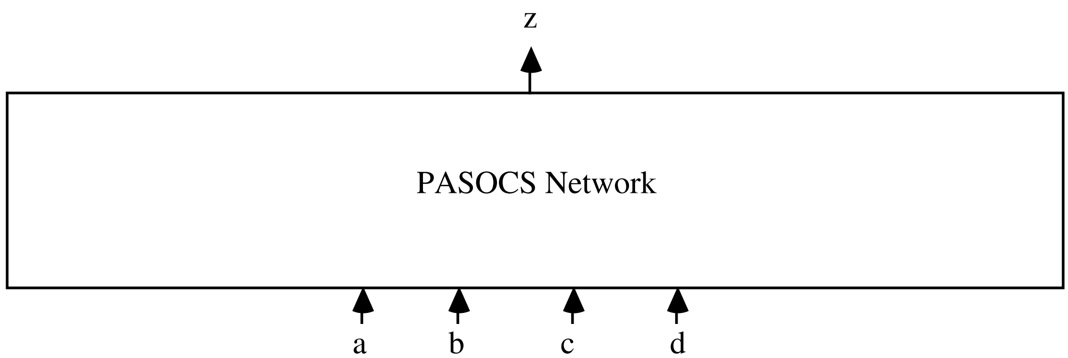

Figure 1 - PASOCS black box

Nodes and Rules: The black box of figure 2 contains nodes connected by a binary tree structure. Each node represents a rule. A rule is represented as a 3-tuple. The first element is a conjunction of boolean variables (the left hand side of the rule), the second element is the rule output (1 or -1), and the third element is the priority of the rule. The priority is a positive integer value with 1 being the lowest priority. The rule of node 1 in figure 2 has priority 2. It says to output (z=-1) whenever (a=1) and (b=1) and (c=0). Node 4 is currently empty.

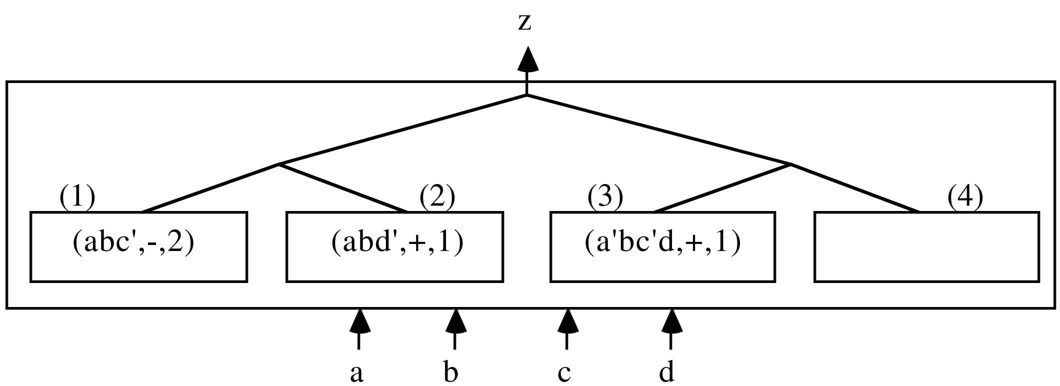

Figure 2 - Internal node representations

Execution: When the black box of figure 3 receives the input (a=1, b=1, c=0, d=0), it broadcasts these values down the binary tree structure to each node. Each node independently and in parallel compares itself to the input. If the input matches the rule, then the node passes up the binary tree a message consisting of the rule output and the rule priority. If the node does not match, no message is sent. *When two messages arrive at a junction in the tree, the one with higher priority (larger integer) is passed on while the other is discarded.* In the case of figure 3 and the given input, nodes 1 and 2 both fire. Since node 1 has priority 2, and node 2 has priority 1, the system outputs the result of node 1, which specifies a output of (z=-1).

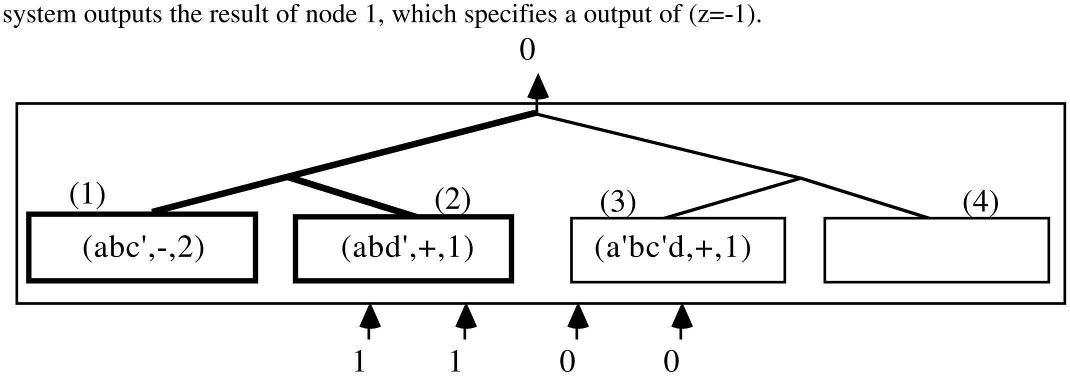

Figure 3 - PASOCS during input processing

Learning : When a new rule is presented to the system, it is broadcast down the tree to the nodes. Each node compares itself with the rule, and depending on how it matches, the node initiates an appropriate operation.

The matching and operations occur independently and in parallel. These operations make up the learning algorithm. We present some examples to give a flavor for the basic learning algorithm. In the following, assume we wish to add the rule bd' => z, to the black box of figure 3. The following actions would take place. Though we present them sequentially, they would take place in parallel in an actual system.

Node Addition and Consistency Resolution: To resolve inconsistencies the system need only give the new rule a priority one greater than any rule it contradicts. The rule bd' => z is broadcast and all nodes which contradict the new rule (i.e. nodes which could be simultaneously matched for some input, but would cause the opposite output) send their priority up the tree. The highest priority arrives at the top and the new rule is added to the network with priority one higher than the highest inconsistent priority. In the case of figure 3, only node 1 (abc' = >z') is inconsistent with bd' = >z and it has priority 2. The system, therefore, adds bd' => z with priority 3, as shown in figure 4.

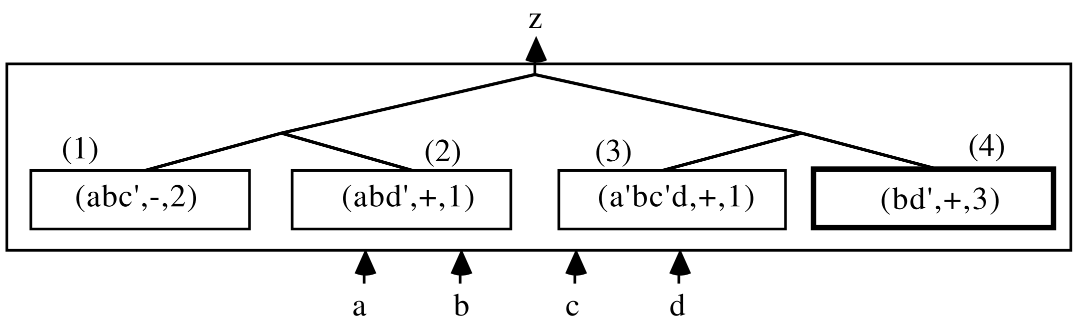

Figure 4 - Adding a new rule

Generalization: The new rule bd' => z has made node 2 redundant, since it will be matched in all cases where node 4 is matched. As part of the learning algorithm, the redundant node 2 deletes itself, as shown in figure 5.

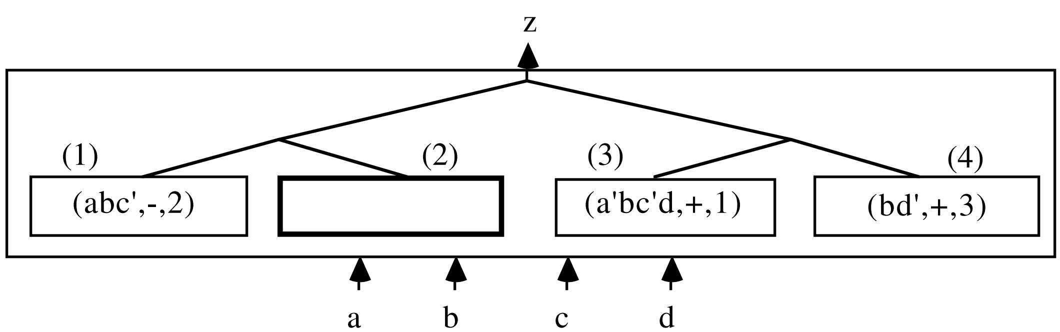

Figure 5 - After self deletion

The new rule bd' => z makes it possible for node 3 of figure 5 to generalize itself by removing the variable d. As part of the learning algorithm, node 3 minimizes itself, as shown in figure 6.

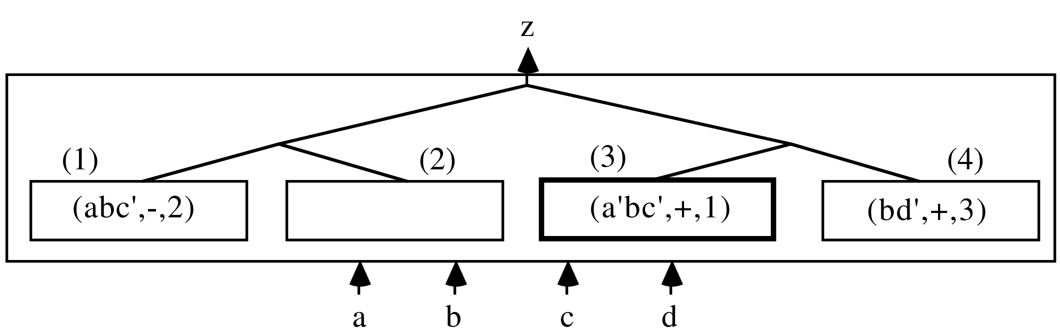

Figure 6 - After further generalization

This gives a quick view of the style of learning mechanisms available in a PASOCS system. We now conclude the list of PASOCS attributes begun earlier:

- In Prioritized ASOCS (PASOCS), whenever a new rule is learned, at most one node need be added to the system.
- PASOCS is guaranteed to learn any set of valid rules.
- PASOCS need only see a rule once when learning a rule set.
- PASOCS does not unlearn old rules when learning new rules, except in the case of conflict resolution.
- PASOCS can support multiple output variables. In this case a node has a different priority and output value for each output variable, allowing efficient sharing of nodes.

Due to the localist node-per-rule nature of PASOCS, each node is independent of every other node. Thus any network topology can be chosen to best fulfill such needs as cost, speed, and fault tolerance. The particular topological instantiation proposed above is that of a binary tree. Due to this decision, we can state that the PASOCS:

- processes inputs in $O(log(n))$ time, where n is the number of nodes (rules) in the network;
- learns individual rules in $O(log(n))$ time, where n is the number of nodes (rules) in the network.

Current Extensions: Flexible Generalization

With the ability of guaranteed fast learning, the real question then concerns the ability to generalize on inputs outside of the training set. Current ASOCS models generalize by seeking critical variables and critical relative groupings of variables in order to generalize higher order functions. Generalization, however, is a very application dependent phenomenon, and no single style of generalization is best for all problems.

We are currently adding to ASOCS the ability to support flexible and parameterizable generalization. This makes possible the new paradigm of a system which *seeks to find the best style(s) of generalization for the specific problem during the learning phase in an adaptive and self-organizing fashion.* Flexible generalization is accomplished by augmenting the standard ASOCS generalization mechanisms with methods from AI and pattern recognition, together with novel schemes. These include:

- Hamming distance (variable mismatches).
- Percentage mismatch.
- Hamming with weighted variables (analog weights).
- Weighted sets of discovered critical features (higher order variable groupings).
- Confidence level (CL) at nodes: this would be as simple as an analog weight which changes in a non-linear way as a function of inputs which contradict or reinforce the feature represented by the node. The CL is not only used as part of the generalization function, but as part of an overall output representing the confidence of the network in the current generalization. Two important side effects of using the CL in ASOCS are:
    a) Ability to handle noisy training sets.
    b) Ability to do standard training set learning along with the normal ASOCS incremental learning.
- Direct instance and prototype generalization using both novel and AI techniques of structural prototype generalization.

Efforts are currently underway to implement these generalization paradigms (and their hybrids) into ASOCS, with a parallel study of actual applications in order to understand the types of generalization most critical to support real world problems.

Current Extensions: Automatic Feature Extraction

A second major effort in this research deals with automatic discovery of pertinent input features. Many times input to a system comes in analog form. This can be handled either by direct analog manipulation (as in many conventional neural networks or in an ASOCS with confidence levels), or by coarse coding the analog input into a set of discrete values. This coarse coding appears to be ubiquitous in natural decision making systems where such analog variables as temperature are broken into coarse categories as cold, warm, and hot. The specific coding for the feature seems to be non-constant and depends on the particular function to which it is input. We apply a similar scenario to learning in an ASOCS system.

Current ASOCS support multi-state variables very efficiently. Assume an input to a dichotomizing ASOCS called MaxVoltage, which is an analog variable. In many cases such a variable, when seen as one feature in a large training set, naturally occurs in a small finite set of clusters, rather than in a smooth continuum. By coarse coding the variable within its natural clusters we can represent the variable in specific training examples such as

MaxVoltage:A & X => 1,
MaxVoltage:C & Y => 0,
etc.,

where MaxVoltage is the multi-state variable, {A, B, C, etc.} are the set of possible states of the variable (the clusters), & is the *logical and*, and X and Y represent arbitrary sets of other features. (i.e. The first example instance above states that if the MaxVoltage feature falls in category A, and all the other features of X are currently matched, then the output should be 1). This coarse-coding can be done automatically with a spontaneous learning mechanism. The current system uses an unsupervised competitive learning system with dynamically allocatable output nodes as a preprocessor to an ASOCS. The competitive learner maps raw analog data into multi-state data which is then input to the fast learning supervised ASOCS.

However, in most cases, a competitive learner can only make a guess at one possible reasonable categorization of the raw input. The important extension that we propose is to *guide the spontaneous learning mechanism with feedback from the supervised ASOCS so that the competitive learner can dynamically adjust its categorizations to support improved learning and generalization*. Consider two examples of how this could be done.

Assume that seemingly contradictory examples of the form

MaxVoltage:A & X => 1
MaxVoltage:A & X => 0

are commonly encountered, where X is an arbitrary set of features In this case, discrimination between the examples may not be possible because of insufficient features or improperly categorized features. The ASOCS will then feedback the information that MaxVoltage:A may not be properly clustered. The competitive learner, in turn, can adjust a local threshold to break MaxVoltage:A into 2 or more categories, leading to potential discrimination.

As another mechanism, assume that examples of the form

MaxVoltage:A & X => 1
MaxVoltage:B & X => 1

are commonly encountered. In this case feedback from the supervised ASOCS can guide the competitive learner to adjust a local threshold such that MaxVoltage:A and MaxVoltage:B are clumped into a single category, since the separate categories appear to have no discriminating ability.

This style of interaction between supervised and unsupervised learning system leads to a powerful emergent model which can take advantage of the strong points of both paradigms.

Besides the two areas just presented, other ASOCS research includes physical realization, extension of data and functional primitives, and integration with other computational paradigms.

Bibliography

1. Chang, J. and J. J. Vidal, Inferencing in Hardware, *Proceedings of the MCC-University Research Symposium*, Austin, TX, (July 1987).

2. Hughes, B., Prioritized Rule Systems, MS Thesis, Brigham Young University, 1990.

3. Martinez, T. R., Adaptive Self-Organizing Logic Networks, PhD Dissertation, University of California Los Angeles, Los Angeles, California, 1986.

4. Martinez, T. R., Adaptive Self-Organizing Concurrent Systems, *Progress in Neural Networks*, **vol. 1**, Ch. 5, 105-126, Ablex Publishing, 1990.

5. Martinez, T. R. and Vidal, J.J., Adaptive Parallel Logic Networks, *Journal of Parallel and Distributed Computing*, **vol. 5**, 26-58, 1988.

6. Martinez, T. R. and Campbell, D. M., A Self-Adjusting Dynamic Logic Module, *Journal of Parallel and Distributed Computing*, **vol. 11**, no.4, pp. 303-313, 1991.

7. Martinez, T. R. and Campbell, D. M., A Self-Organizing Binary Decision Tree for Incrementally Defined Rule Based Systems, *IEEE Transactions on System, Man, and Cybernetics,* **vol. 21**, 1991.