M. A. Sartori and P. J. Antsaklis, "Neural Computing and Production Systems," P roc. o f t he 3 rd I EEE I nternational S ymposium o n I ntelligent C ontro l , pp. 665-670, Arlington, VA, Aug. 24-26, 1988.

NEURAL COMPUTING AND PRODUCTION SYSTEMS*

Michael A. Sartori and Panos J. Antsaklis
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

The application of neural computing to the problem of matching in production systems is addressed. The computation time required by this problem can be significantly reduced by using the massive parallelism and pattern recognition capabilities available through neural computing. A new neural computing model, called here the ProNet, is introduced and explained in detail. The ProNet is applied to the match phase of the production system interpreter in an attempt to yield a reduction in time and space requirements by matching all of the productions to all of the working memory elements simultaneously.

1.0 INTRODUCTION

The production system, a special type of expert system, will probably continue to be used to assist both humans and computers in specific tasks for future applications of artificial intelligence to Intelligent Control. If this type of system is to be used efficiently in real-time applications, the speed at which the production system operates must be considered. Present production system schemes are becoming faster, but still can be improved. A new alternative approach to present production system schemes uses neural computing to increase the speed of the production system. This is accomplished by performing the match phase of the developed production system using special hardware.

Production systems are expert systems which use rules, called *productions* (*rules, production rules*), to represent knowledge and which use a particular interpreter to perform the actions of the production system. The form for the production addressed in this paper is

IF a_1 and a_2 and ... and a_j THEN b_1 and b_2 and ... and b_k

where the a_i are the *antecedents* and the b_i are the *consequences* of the particular production. Customarily, the conjunction of the antecedents is referred to as the left-hand side (LHS), and the conjunction of the consequences is referred to as the right-hand side (RHS). The *working memory* (WM) contains the data which is compared to the productions. The individual elements of the WM are referred to as the *working memory elements* (WMEs). The *production interpreter* performs the comparison of the WM to the productions. It is commonly assumed that the interpreter should have a three phase cycle:

(i) *Match.* Compare the LHS of all of the productions to the WMEs. If the LHS is satisfied, include the production in the *conflict set*, the set of satisfied productions for the present WM state.

(ii) *Select.* Choose one production from the conflict set to execute.

(iii) *Act.* Execute the production in accordance with the RHS of the chosen production.

Of the three phases, the match phase traditionally consumes the most time of the production interpreter. Using conventional approaches, a production interpreter can spend more than 90% of its time in the match phase of the production cycle [1]. The Rete Match Algorithm, introduced in [1,2], avoids the brute force approach ofsequentially matching productions against WMEs by manipulating the productions and the WMEs to form a software tree structure to increase the speed of the production interpreter. Since its introduction, other Rete based algorithms which attempt to increase the speed of the production interpreter have been introduced [3-10]. To further reduce the amount of time consumed by the interpreter in the match phase, special hardwares have been developed using parallelism and multiprocessor architectures [6,11-18]. Almost all of these attempts are based, at least in part, on the Rete Match Algorithm, which is assumed to yield the most efficient match phase of the production system interpreter. These architectures strive to decrease the time required in the match phase by attempting to match as many rules as possible in parallel and by attempting to fire as many rules as possible in parallel. In addition, because these proposed architectures intend to use a multiprocessor implementation, they will consume a significant amount of physical space when realized. A new method is proposed here which simultaneously matches all of the productions to all of the WMEs in parallel via neural computing.

In this paper, the potential for using neural computing for the match process of the production system interpreter is investigated. The use of neural computing to aid expert systems was addressed in [19] which proposed to increase the speed of expert systems by using neural computing for the select phase. The use of neural computing in this paper attempts to achieve increased speeds and reduced space requirements by using neural computing for the match phase. Using a new neural computing model, the ProNet, which resembles the single layer perceptron of [23], to perform the computations in the match phase of the production interpreter, the amount of time required in the match phase can be reduced. Compared to the Rete Match Algorithm, the proposed method achieves a significant increase in speed. In addition, with the recent advances in the realization of very large scale integration and electro-optical techniques for analog and parallel computation, the possibility exists to reduce the physical space required by the production system.

In Section 2, a new model for neural computing, the ProNet, is introduced, examined in detail, and, in Section 3, used to perform the match phase for the interpreter of a developed production system. Next, in Section 4, a simulation of a small production system using the ProNet is presented. Finally, in Section 5, some concluding remarks are made.

2.0 THE PRONET

The ProNet is a new neural computing model based on another neural computing model, the Hamming net (HN). A derivation and explanation of the HN can be found in [20,21]. The HN is a combination of a feedforward net and a feedback net and is used to classify patterns for speech recognition. The HN is given an input vector and specifies which stored *exemplar pattern* it most closely matches as a function of the Hamming distance. The exemplar patterns are previously stored patterns which exemplify all of the possible patterns to be passed through the system. Due to the fact that only one exemplar patten can be identified with the HN, the "ProNet" is created here and used to identify more than one exemplar pattern. The ProNet uses the HN's feedforward perceptron-like net with slight alterations to the weights and replaces the HN's feedback maxnet by changing the biases of the feedforward net. The ProNet's purpose is for pattern identification in production systems.

The ProNet, shown in Figure 2.1, is a feedforward net closely resembling and mimicking the operation of the single layer perceptron. An input vector is presented to the ProNet through the input vector $u = [u_1\ u_2\ \ldots\ u_N]$. The ProNet is expected to identify which exemplar patterns $x \in X$ appropriately match it, where X is the set of all exemplar patterns. In the ProNet, the exemplar patterns are stored via the weights w_{ij}, where the subscript i denotes which input element and the subscript j denotes which exemplar pattern. The output vector $y = [y_1\ y_2\ \ldots\ y_M]$ identifies which exemplar patterns have been matched. Basically, the ProNet simultaneously compares in parallel an input vector u to every exemplar pattern $x \in X$.

One advantage in using the ProNet is the ease with which the weights and the biases are found, compared to the single layer perceptron which requires multiple passes of the input data to train the weights. The weights w_{ij} and the biases c_j are determined based on the exemplar patterns. Assume all of the input vectors to the ProNet

*This work was partially supported by the Jet Propulsion Laboratory, Pasadena, California under contract 957856.

0-8186-2012-9/89/0000/0665$01.00 © 1989 IEEE

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.

M. A. Sartori and P. J. Antsaklis, "Neural Computing and Production Systems," P roc. o f t he 3 rd I EEE
I nternational S ymposium o n I ntelligent C ontro l , pp. 665-670, Arlington, VA, Aug. 24-26, 1988.

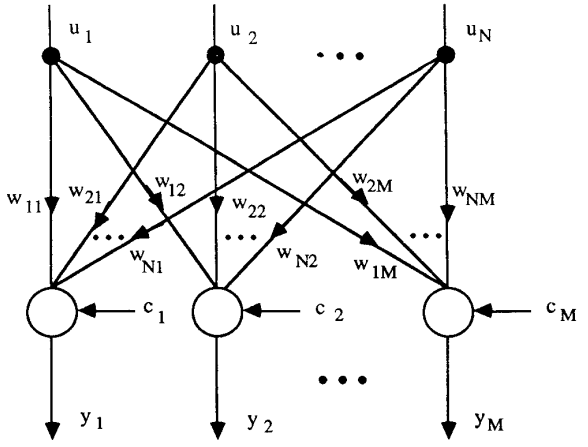

Figure 2.1 The ProNet.

are binary signals with a fixed length N having elements valued either one (HI) or zero (LO). Assume also there are M exemplar patterns, which are to be "stored" in the M nodes of the net. Then, the weights and biases are chosen according to equations (1) and (2).

$$w_{ij} = \begin{cases} 1 & \text{if } x_{ij} = 1 \\ 0 & \text{if } x_{ij} = 0 \end{cases} \qquad 1 \le i \le N,\ 1 \le j \le M \qquad (1)$$

$$c_j = \Delta - \sum_{i=1}^{N} x_{ij} \qquad 1 \le j \le M,\ 0 < \Delta < 1 \qquad (2)$$

The value w_{ij} is the weight for the i^{th} input element to the j^{th} node. The value x_{ij} is the i^{th} element of the j^{th} exemplar stored. The value c_j is the bias to be added to the j^{th} node. The summation in the bias c_j is the total number of HI elements in the j^{th} exemplar pattern. The value Δ of c_j is some constant between zero and one, but should be the same for each bias c_j. The reason for including the value Δ in the bias will be explained later in this section. Note that all of these values are constant once they are determined; there is no need to adapt to new ones or change them during the operation of the ProNet.

Once the weights w_{ij} and the biases c_j are determined, the ProNet is prepared for operation. A binary pattern is presented to the ProNet as the input vector u with N elements. The output of the ProNet is shown in equation (3).

$$y_j = f\left(\sum_{i=1}^{N} w_{ij}\, u_i + c_j \right) \qquad 1 \le j \le M \qquad (3)$$

The value y_j is the j^{th} element of the output vector y. For each output element y_j, the N elements u_i of the input vector u are multiplied by the weights w_{ij}, summed with the bias c_j, and passed through the threshold logic nonlinearity $f(a)$ illustrated in Figure 2.2. Note that the sum in equation (3) totals the number of HI elements of the input vector u which coincide with the HI elements of the j^{th} exemplar pattern.

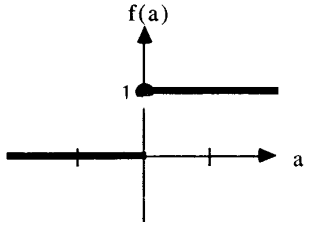

Figure 2.2. Threshold logic nonlinearity.

Substituting in the value for c_j, equation (3) becomes equation (4).

$$y_j = f\left(\sum_{i=1}^{N} w_{ij}\, u_i - \sum_{i=1}^{N} x_{ij} + \Delta \right) \qquad 1 \le j \le M,\ 0 < \Delta < 1$$

$$= \begin{cases} f(\Delta) & \text{if } \forall\ x_{ij} = 1,\ \exists\ u_i = 1 \\ f(\gamma + \Delta) & \text{else, where } \gamma = \sum_{i=1}^{N} w_{ij}\, u_i - \sum_{i=1}^{N} x_{ij} \le - \end{cases}$$

$$= \begin{cases} 1 & \text{if } \forall\ x_{ij} = 1,\ \exists\ u_i = 1 \\ 0 & \text{else} \end{cases} \qquad (4)$$

When all of the HI elements of the input vector u coincide with the HI elements of the exemplar pattern, the summation of the weighted inputs and bias will be positive, i.e. Δ, and after passing through the nonlinearity, the output y_j will be HI. Otherwise, the summation will be negative, and after passing through the nonlinearity, the output y_j will be LO. If the value Δ is less than or equal to zero, the sum will never be positive. If the value Δ is greater than or equal to one, the sum could be positive or zero when all of the HI elements do not coincide. Thus, the value Δ is a constant between zero and one.

Note that after passing through the function f, all values of a less than zero will become zero and all values of a greater than or equal to zero will become one. Each output y_j of the ProNet corresponds to the j^{th} exemplar patten. If the HI elements of the j^{th} exemplar pattern coincide with the same HI elements of the input vector u, i.e. if $\forall\ x_{ij} = 1,\ \exists\ u_i = 1$, the output y_j will be HI; otherwise, the output y_j will be LO. Since the ProNet compares the input vector u and exemplar patterns $x \in X$ in this manner, more than one output y_j of the ProNet can become HI for a given input vector u.

3.0 THE PRONET AND THE PRODUCTION SYSTEM'S MATCH PHASE

Because of the massive parallelism and the pattern recognition capabilities available, the ProNet becomes an excellent tool to reduce the amount of time required in the match phase of the production interpreter. To accomplish this, the match problem needs to be properly handled to use the ProNet and its properties. To implement the match process of the production interpreter using the ProNet, the LHSs of the productions are used as the exemplar patterns, and the WMEs are used as the input elements. A parallel simultaneous match of all of the LHSs to all of the WMEs is performed.

The performance of the match process is degraded by neither the number of productions nor the number of WMEs. The ProNet can also easily cope with large changes to the WM; it is not dependent upon temporal redundancy as the Rete Match Algorithm is. However, note that the proposed use of the ProNet in the match phase is for developed production systems and not for the development of production systems. It is a tool to be used in the implementation of production systems to perform the match more efficiently.

First, the exemplar patterns for the ProNet need to be found. Productions of the form

IF a_1 and a_2 and ... and a_j THEN b_1 and b_2 and ... and b_k

are considered. An assumption is made that all of the productions can be placed in the above form, where a_i and b_i are, respectively, symbolic antecedents and symbolic consequences (time-varying numeric data is not considered here). Once all of the productions of the production system have been found, their LHSs are compared, and the set A of all possible antecedents is established. Note that the set A is also the set of all possible WMEs. Assume there are N possible antecedents and M productions. Each possible antecedent, element of A, is assigned an element of the input vector u of the ProNet. The M exemplar patterns are formed by comparing the LHSs of the productions to the set A of possible antecedents. Thus, the set A has N elements and the set X has M elements. A one (HI) is assigned to the element of the exemplar pattern which coincides with

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.

M. A. Sartori and P. J. Antsaklis, "Neural Computing and Production Systems," P roc. o f t he 3 rd I EEE
I nternational S ymposium o n I ntelligent C ontro l , pp. 665-670, Arlington, VA, Aug. 24-26, 1988.

the appropriate antecedent of the production; a zero (LO) is assigned to the element otherwise. For example, if the set of all possible antecedents is $A = \{a_1, a_2, a_3, a_4\}$, let a_1 be assigned to the first element u_1 of the input vector u, a_2 assigned to the second element u_2, a_3 assigned to the third element u_3, and a_4 assigned to the fourth element u_4. If the first production given is

IF a_1 and a_3 THEN a_2

the exemplar pattern x_1 associated with the first production is [1 0 1 0], where x_{11} is 1, x_{21} is 0, x_{31} is 1, and x_{41} is 0. The weights w_{ij} and the biases c_j are chosen according to equations (1) and (2) using the newly formed exemplar patterns. For the above example, the weight w_{11} is 1, w_{21} is 0, w_{31} is 1, and w_{41} is 0. The bias c_1 is Δ - 2. Thus, the input vector corresponds to the set A of all possible antecedents with each of the N input elements corresponding to one of the N predetermined antecedents. Each of the M exemplar patterns corresponds to one of the M productions.

Once the weights and biases of the ProNet are set, the net is ready for operation. A WM state of the production system is found, and an appropriate input vector u for the ProNet is determined. The WMEs of the WM state correspond to specific members of the set A of possible antecedents and also correspond to particular elements of the input vector u of the ProNet. An input vector u is determined by assigning a HI to the elements of the input vector which correspond to the WMEs of the WM state, and a LO to the elements of the input vector which do not correspond to the WMEs of the WM state. For the above example, if the present WM state contains the WMEs a_1 and a_4, the input vector u is [1 0 0 1]. The newly determined input vector u is then passed through the ProNet. The output vector y immediately identifies which production's LHSs match the WM state. A HI on the output element y_j signifies that the j^{th} production is firable from the present WM state, and a LO on the output element y_j signifies that the j^{th} production is not firable from the present WM state. Finally, the conflict set is formed by gathering all of the productions identified by the HI elements of the ProNet's output vector y. This completes the match phase of the production interpreter.

The cycle of the production interpreter using the ProNet in the match phase can be viewed as a loop of function blocks, as illustrated in Figure 3.1.

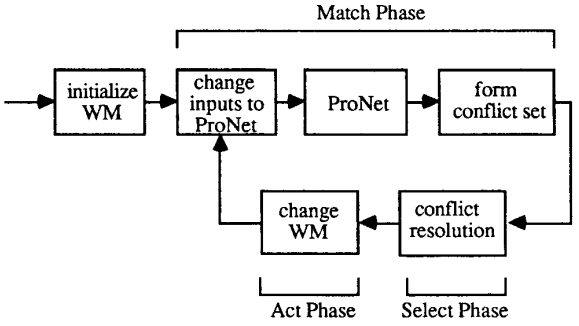

Figure 3.1 Function block diagram for the production interpreter cycle using the ProNet.

For the function block denoting the changes to the inputs of the ProNet, conventional processing needs to be used. Each time the WM is changed, the input vector u of the ProNet needs to be changed. Instead of changing the entire input vector u each time, the changes to the WM performed in the act phase can be saved, and these changes can be used to change the appropriate elements u_i of the input vector u in the first function block of the match phase. For the above example, if the previous WM state is a_1 and a_4 and if the new WM state is a_1 and a_3, the WME a_4 is deleted from the WM and the WME a_3 is added to the WM. These two changes are saved from the act phase. For the match phase, these changes are processed by setting the ProNet input element u_4 LO and the input element u_3 HI. Thus, the previous input vector u is [1 0 0 1] and the new input vector u is [1 0 1 0].

For the function block denoting the ProNet, the actual computations involved in the match phase are performed using ProNet. The computation time here is dependent on neither the number of productions, M, nor the number of WMEs, N. All of the productions represented as exemplar patterns in the ProNet are compared in parallel. So, the number of productions does not affect the computation time. All of the WMEs represented as elements u_i of the input vector u of the ProNet are compared to the productions in parallel. So, the number of WMEs does not affect the computation time.

For the function block denoting the formation of the conflict set, conventional processing needs to be used. The output vector y of the ProNet represents the conflict set. The elements of the vector which are HI represent productions which are firable, and elements which are LO represent productions which are not firable. Conventional processing is used to transform this vector into a form usable for the conflict resolution of the act phase.

In addition to considering production systems with productions having symbolic antecedents and symbolic consequences, production systems which allow productions to contain negated symbolic antecedents and variable symbols as antecedents and consequences can also be considered. If a production contains a negated symbolic antecedent in its LHS, the production can not be fired if that particular antecedent is contained in the WM. The ProNet models the negation antecedent by not allowing the production to become part of the conflict set if the particular antecedent becomes part of the WM. This is accomplished by modifying the weights w_{ij} in the ProNet. Equation (1) now becomes equation (5).

$$w_{ij} = \begin{cases} 1 & \text{if } x_{ij} = 1 \\ 0 & \text{if } x_{ij} = 0 \\ -1 & \text{if negation of } x_{ij} \end{cases} \qquad 1 \le i \le N,\ 1 \le j \le M \qquad (5)$$

The weight w_{ij} is assigned a value of -1 when the j^{th} exemplar patten requires the negation antecedent property for the i^{th} antecedent of the set A of possible antecedents. For the above example with $A = \{a_1, a_2, a_3, a_4\}$, if the first production given is

IF a_1 and a_3 and NOT a_4 THEN a_2

the exemplar pattern x_1 is [1 0 1 0] where x_{11} is 1, x_{21} is 0, x_{31} is 1, and x_{41} is 0. The weight w_{11} is 1, w_{21} is 0, w_{31} is 1, and w_{41} is -1. When the i^{th} antecedent is absent from the WM, the i^{th} input to the ProNet is LO and the operation of the ProNet is the same as before. When the i^{th} antecedent is present in the WM, the i^{th} input to the ProNet becomes HI and the summation of equation (3) for the j^{th} production with the negation antecedent property can no longer be greater than zero. The output y_j becomes zero and the j^{th} production with the negation antecedent is not included in the conflict set. Continuing the above example, if the input vector u is [1 0 1 0], the output vector y_1 is 1. If the input vector u is [1 0 1 1], the output vector y_1 is 0.

Using the ProNet, production systems which contain productions with variable symbols can also be considered. If a production contains a variable symbolic antecedent in its LHS, the production's LHS can match a number of different antecedents and the production is firable from several WM states. Assume that the productions with variable antecedents can match only a finite number of definable antecedents. Thus, new productions are made from the original production through the instantiation of the variable. The new productions' LHSs contain the LHS of the original production with the variable replaced by one of the possible antecedents, and the new productions' RHSs are the same as the RHSs of the original production. If the RHS of the production contains the variable as well, the variable on the RHS is replaced by the same antecedent which replaced the variable on the LHS. Thus, productions which contain variables in the antecedents are instantiated for all possible cases, and a set of new productions is formed without any variables. Exemplar patterns can now be determined from the newly formed productions, and the ProNet's initialization and operation can continue as previously described. For the above example with $A = \{a_1, a_2, a_3, a_4\}$, if the first production given is

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.

M. A. Sartori and P. J. Antsaklis, "Neural Computing and Production Systems," P roc. o f t he 3 rd I EEE I nternational S ymposium o n I ntelligent C ontro l , pp. 665-670, Arlington, VA, Aug. 24-26, 1988.

IF a_1 and v THEN a_2

the variable symbol is v which can assume three different symbolic values: a_2, a_3, and a_4. So, the first production is instantiated and three new productions are formed from the first one.

IF a_1 and a_2 THEN a_2
IF a_1 and a_3 THEN a_2
IF a_1 and a_4 THEN a_2

These newly formed productions are used to determine the exemplar patterns for the ProNet as described previously. It should be noted that the instantiation of all of the variables for the developed production system and the forming of the new set of productions are performed before the ProNet's dimensions, weights, and biases are set. This pre-processing of the developed production system can be accomplished using conventional processing and could possibly require a large amount of time depending on the number of variables and the number of instantiations per variable in the developed production system.

4.0 A SIMULATION OF A PRODUCTION SYSTEM USING THE THE PRONET

The ProNet is applied to the match phase of a simple production system. The list of productions used is shown in Figure 4.1.

(A B) (A D) (C D) (E Π)	→	(A D) (B E) (C E) (D A)
¬(A B) (AC) (Π B) (Π D)	→	(A D) (C E) (D Π)
(A B) (A E) ¬(C D) (D B) ¬(E D)	→	(A B) (B C) (C D)
¬(A B) (B D) (B E)	→	(A B) (B C) (C D)
(Π D) (Π E) (D Γ) (Π Γ)	→	(Π E) (B C) (Γ Π)
¬(A B) (C D) (D A) (D E)	→	(A D) (A E) (B E) (E D)
¬(A D) (A E) (C A) ¬(C D) (D E)	→	(A C) (A E) (D E) (E D)
(Π B) (Π D) (B Π) (C E)	→	(C A) (D Π) (E C)
(B D) (B E) ¬(D E) (E B)	→	(B D) (C A) (C B) (C D) (CE)
(A D) (B C) (D C) (D Π)	→	(B A) (B C) (B E) (Π D)

Figure 4.1 Productions used for the simulation.

This simple production system is of the particular form considered in this paper. For example, the second production can be written as

IF NOT (A B) and (A C) and (Π B) and (Π D)
THEN (A D) and (C E) and (D Π)

The format for the productions is similar to the examples used in [23]. There are ten productions with the LHS and the RHS separated by the implication symbol →. The antecedents and consequences are pairs comprised of the first five letters of the English alphabet and designated by two letters enclosed in a set of parentheses. The LHSs and RHSs are conjunctions of the antecedents and conjunctions of the consequences, respectively. The only rule is that no pair may have repeated letters, e.g. the pair (B B) is not allowed. Besides the letter symbols, the production system also consists of negations and variables. The negation antecedents are denoted by the pairs with the not symbol ¬ in front. The antecedents and consequences with symbolic variables are denoted by the pairs with the Greek symbols Π and Γ.

Before the ProNet's weights and biases can be found, the variables must be addressed. As stated in Section 3, the variables are assumed to only represent a finite number of definable antecedents. Thus, each variable is instantiated for all possible instances and a new set of productions results. For example, the fifth production of Figure 4.1 is expanded into the nine new productions shown in Figure 4.2. The variable Π of the fifth production can assume the symbols A, B, and C, and the variable Γ of the fifth production can assume the symbols A, B, C, and E. When this is done to each production with a variable, the new production system has twenty-seven productions instead of ten.

The LHSs of the twenty-seven newly formed productions are now used to form the exemplar patterns for the ProNet. First, the total number N of possible antecedents needs to be found. By counting the different possible pairs, there are twenty possible

(A D) (A E) (D B) (A B)	→	(A E) (B C) (B A)
(A D) (A E) (D C) (A C)	→	(A E) (B C) (C A)
(A D) (A E) (D E)	→	(A E) (B C) (E A)
(B D) (B E) (D A) (B A)	→	(B E) (B C) (A B)
(B D) (B E) (D C) (B C)	→	(B E) (B C) (C B)
(B D) (B E) (D E)	→	(B E) (B C) (E B)
(C D) (C E) (D A) (C A)	→	(C E) (B C) (A C)
(C D) (C E) (D B) (C B)	→	(C E) (B C)
(C D) (C E) (D E)	→	(C E) (B C) (E C)

Figure 4.2 The fifth production with its variable instantiated for all possible instances.

antecedents; the set A has these twenty elements. Next, each possible antecedent needs to be assigned to an input element of the ProNet. Here, the possible antecedents are assigned to the elements in an ordered fashion as shown in Figure 4.3.

Antecedent	Input Element
(A B)	1
(A C)	2
(A D)	3
•	•
•	•
•	•
(E C)	19
(E D)	20

Figure 4.3 Assigning of the antecedents to the ProNet input elements.

The exemplar patterns of each production are formed by assigning a one (HI) to elements which coincide to the antecedents of the LHSs and assigning a zero (LO) to elements otherwise. For example, from Figure 4.2, the first production's exemplar pattern x_1 is

[1 0 1 1 0000 0000 0100 0000]

If this is done for each production, there would be twenty-seven exemplar patterns with twenty elements each.

The weights and biases of the ProNet are now computed using equations (5) and (2). As described previously, the weights corresponding to the elements of exemplar patterns with negation antecedents are assigned a value of -1. For example, from Figure 4.1, the seventh production's LHS produces the following weights

[0 0 -1 1 0000 1 0 -1 0 0001 0000]

For the biases, a value of .5 was chosen for the constant Δ. Once the weights and biases are found, the ProNet is ready for operation.

For the operation of the production system, the match phase was performed using the ProNet. For the select phase, the conflict resolution was performed by prioritizing the productions and choosing the production in the conflict set with the highest priority. The productions were prioritized by assigning the first production of Figure 4.1 the highest priority and the rest of the productions lower priorities in a descending fashion. Note that to use the ProNet for the match phase, this type of conflict resolution does not need to be used. For example, other strategies for conflict resolution can be found in [23]. The act phase was performed by deleting the chosen production's antecedents from the WM and adding the chosen production's consequences to the WM. This results in switching elements of the ProNet's input vector, as described previously.

For the simulation, the initial WM state considered is

(A B) (A C) (A D) (B C) (B D) (B E) (C B) (C E) (D B) (D C) (D E) (E D)

which results in the following initial input vector u for the ProNet

[0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 0 0 1]

Figure 4.4(a) illustrates the input to the ProNet for the first WM state. Figure 4.4(b) illustrates the output of the ProNet for the first WM state. There are eight productions in the conflict set, and the production interpreter selects the one with the highest priority, production eight. The WM is changed by deleting and adding the appropriate antecedents; the second WM state is formed. Figure 4.5 illustrates the input and output for the ProNet for the second WM state. There are ten productions in the conflict set. If this were continued, the production interpreter would select the one with the

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.

M. A. Sartori and P. J. Antsaklis, "Neural Computing and Production Systems," P roc. o f t he 3 rd I EEE
I nternational S ymposium o n I ntelligent C ontro l , pp. 665-670, Arlington, VA, Aug. 24-26, 1988.

highest priority, production four. The WM would be changed by deleting and adding the appropriate antecedents, and the third WM state would be formed.

5.0 CONCLUDING REMARKS

Using neural computing via the ProNet, the time required by the match phase can be considerably reduced and thus, the overall time required by the production interpreter can be decreased. The cycle of the production interpreter employing the ProNet in the match phase can be viewed as a loop of function blocks, as illustrated in Figure 3.1. Examining each of the blocks for the match phase, the time required for the match phase can be quantified.

For the function block denoting the changes to the inputs of the ProNet, additions and deletions to the WM need to be considered. Assume that the changes made to the WM in the act phase are saved and are used to change the ProNet's input elements in the match phase. If WMEs are added according to the RHS of the fired production and if WMEs are deleted according to the LHS of the fired production, the changes made to the inputs of the ProNet by conventional processing are a function of the number of antecedents and the number of consequences in the fired production. Thus, the time required to change the inputs to the ProNet is on the order of the number of antecedents in the LHS and the number of consequences in the RHS of the production. In [12], a number of different production systems were examined, and it was found that the typical production contains three to six antecedents and two to four consequences. So, the time required to update the inputs to the ProNet is "relatively" small. Note that if a multiprocessor architecture was used, the changes to the inputs of the ProNet could be performed in parallel, and a further increase in speed could be achieved along with a further increase in physical space used.

For the function block denoting the ProNet, the time required to achieve an output of the ProNet is dependent on the speed of the summation and the nonlinearity of equation (3), which is dependent on neither the number of productions in the system nor the number of WMEs in the current WM state. Given the current state of technology, the time for this block is probably much smaller than the time for the other two blocks of the match phase.

For the function block denoting the formation of the conflict set, the time required is dependent on the manner in which the match phase and the select phase are interfaced. The output vector y of the ProNet is the conflict set with the HI elements representing firable productions. This vector needs to be transformed into a representation useful to the select phase. One possibility for the transformation is to have the output elements y_j, when they are HI, point to the addresses where the productions are stored in a conflict resolution usable form. Whatever method is used for the transformation, a minimal amount of conventional processing will probably be needed because of the form of the ProNet's output vector.

Thus, given a single processor architecture and given that each of the three function blocks of the match phase requires a "relatively" small amount of time, the overall time to conduct the match phase will be "relatively" small. Using the asymptotic complexity theory of [24], Table 5.2 compares results for the ProNet and some of the results for the Rete Match Algorithm in [2].

	ProNet		Rete Match Algorithm	
Complexity Measure	Best	Worst	Best	Worst
Effect of WM size on time for one firing	$O(1)$	$O(1)$	$O(1)$	$O(W^{2A-1})$
Effect of number of productions on time for one firing	$O(1)$	$O(1)$	$O(log_2P)$	$O(P)$

A is the number of antecedents per production
P is the number of productions
W is the number in WMEs

Table 5.2 Time complexity of the match phase using the ProNet.

A cost of $O(g(x))$ denotes that the cost is dominated by the function $g(x)$ for large x. A cost of $O(1)$ denotes that the cost is unaffected by the factor. Comparing the ProNet and the Rete Match Algorithm for the above two criterions, the cost of the match phase using the ProNet is much smaller than the cost of the match phase using the Rete Match Algorithm. Even for the worst case, the ProNet's time for one firing is still unaffected by the WM size and the number of productions.

The above comparison does not include the conventional processing required by the other two function blocks of the match phase. However, the cost to form the input vector u for the ProNet using the changes made in the act phase will probably be on the order of the number of changes to be made to the elements u_i of the input vector u. The cost to transform the output vector of the ProNet into a form usable by the select phase can not be analyzed presently because it is dependent upon the implementation of the select phase. Overall, the match phase using the ProNet is independent of both the number of productions and the number of WMEs and uses hardware to perform the match; the match phase using the Rete Match Algorithm is dependent on both the number of productions and the number of WMEs and uses software to perform the match.

In addition, the above comparison does not include current attempts at increasing the speed of the match phase using a modified Rete Match Algorithm or similar algorithms and/or special hardware. Previous attempts to reduce the time required by the match phase of the production interpreter have been reported in the literature. Using the YES/OPS production system language and advances in the Rete Match Algorithm, a drop in CPU time was reported in [3]. Using partitioning of the productions, a reduction in production cycles was reported in [7,10,8]. The results from the literature of using special hardware are summarized in Table 5.4.

Architecture	WME Changes/Sec	Productions/ Second	Use of modified Rete?	Hardware Needed	Reference
MAPPS	10,000	-	Yes	128 PEs	[13]
CMU-PMS	9400	-	Yes	32 Ps	[6]
PESA-1	25,000	8000	Yes	32 PEs	[14]
DADO	225	85	No (TREAT)	1023 PEs	[16,6]
NON-VON	2000	903	Yes	16032 PEs	[13]
Olfazer's	4500	-	Yes	512 Ps	[6]

- is not available
Ps is processors
PEs is processing elements

Table 5.4 Summarized results of special hardware implementations.

Although actual timing of the ProNet in the match phase has not yet been performed, the authors believe that its performance will be superior to the ones currently available.

In addition, note that the current hardware implementations cited in Table 5.4 use either many processors or many processing elements. This will cause the production system to occupy a considerable amount of physical space. Using a single processor and the ProNet in the match phase, physical space occupied by the production system can possibly be less than ones currently available. With the ProNet in the match phase, neither a many processor architecture nor a many processing element architecture needs to be used. With the use of very large scale integration and/or electro-optical techniques, the layout of the ProNet need not consume a large amount of space. If elements of the input vector of the ProNet are not wired to the summation nodes when the weights between them are zero, a further reduction in the space required can be achieved. Although the actual implementation of the match phase using the ProNet has not yet been performed, the authors believe that, compared to the current ones available, a reduction in the physical space occupied by the production system can be achieved.

ACKNOWLEDGEMENTS

The authors wish to thank Zhiqiang Gao for his comments on the earlier versions of this paper and to especially thank Kevin M. Passino for his valuable discussions, comments, and careful reading of the earlier versions of this paper.

6.0 REFERENCES

[1] Forgy C.L., On the Efficient Implementation of Production Systems, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, February 1979.

[2] Forgy C.L., "Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Problem", Artificial Intelligence, Vol. 19, pp. 17-37, 1982.

[3] Schor M.I., Daly T.P., Ho S.L., Tibbitts B.R., "Advances in Rete Pattern Matching", Proceedings of the 1986 National Conference on Artificial Intelligence, pp. 226-232, 1986.

[4] Stolfo S.J., "Five Parallel Algorithms for Production System Execution on the DADO Machine", Proceedings of the National Conference on Artificial Intelligence, pp. 300-307, 1984.

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.

[5] Miranker D.P., "Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE", Proceedings of the International Conference on Fifth Generation Computer Systems, pp. 449-457, 1984.

[6] Gupta A., Forgy C.L., Newell A., Wedig R., "Parallel Algorithms and Architectures for Rule-Based Systems", Proceedings of the 13th International Symposium on Computer Architecture, pp. 28-37, 1986.

[7] Ishida T., Stolfo S., "Towards the Parallel Execution of Rules in Production System Programs", Proceedings of the 1985 International Conference on Parallel Processing, pp. 568-575, 1985.

[8] Oflazer K., "Partitioning in Parallel Processing of Production Systems", Proceedings of the 1984 International Conference on Parallel Processing, pp. 92-100, 1984.

[9] Moldovan D.I., "A Model for Parallel Processing of Production Systems", Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 568-573, 1986.

[10] Tenorio M.F.M., Moldovan D.I., "Mapping Production Systems into Multiprocessors", Proceedings of the 1985 International Conference on Parallel Processing, pp. 56-62, 1985.

[11] Won H., "On Parallel Processing of Universal Rule Based Expert System", Proceedings of the 1987 Western Conference on Expert Systems, pp. 136-143, 1987.

[12] Forgy C., Gupta A., Newell A., Wedig R., "Initial Assesment of Architectures for Production Systems", Proceedings of the 1984 National Conference on Artificial Intelligence, pp. 116-120, 1984.

[13] Forgy C., Goopta A., "Preliminary Architecture of the CMU Production System Machine", Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences, 1986, pp. 194-200, 1986.

[14] Oshisanwo A.O., Dasiewicz P.P., "A Parallel Model and Architecture for Production Systems", Proceedings of the 1987 International Conference on Parallel Processing, pp. 147-153, 1987.

[15] Schreiner F., Zimmermann G., "PESA 1 - A Parallel Architecture for Production Systems", Proceedings of the 1987 International Conference on Parallel Processing, pp. 166-169, 1987.

[16] Ramnarayan R., Zimmermann G., Krolikoski S., "PESA-1: A Parallel Architecture for OPS5 Production System", Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences, pp. 201-205, 1986.

[17] Stolfo S.J., Miranker D.P., "DADO: A Parallel Processor for Expert Systems", Proceedings of the 1984 International Conference on Parallel Processing, pp. 74-82, 1984.

[18] Gupta A., "Implementing OPS5 Production Systems on DADO", Proceedings of the 1984 International Conference on Parallel Processing, pp. 83-91, 1984.

[19] Shaw D.E., "NON-VON: A Parallel Machine Architecture for Knowledge-Based Information Processing", Proceedings of the Seventh International Joint Conference on Artificial Intelligence, pp. 961-963, 1981.

[20] Castelaz P., Angus J., Mahoney J., "Application of Neural Networks to Expert Systems and Command and Control Systems", Proceedings of the 1987 Western Conference on Expert Systems, pp. 118-125, 1987.

[21] Lippmann R.P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp 4-22, April 1987.

[22] Lippmann R.P., Gold B., Malpass M.L., A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification, Lincoln Laboratory, Massachusetts Institute of Technology, Technical Report 769, May 1987.

[23] McDermott J., Forgy C., "Production System Conflict Resolution Strategies", D.A. Waterman, F. Hayes-Roth (Eds.), Pattern-Directed Inference Systems, Academic Press, New York, 1978.

[24] Aho A.V., Hopcroft J.E., Ullman J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

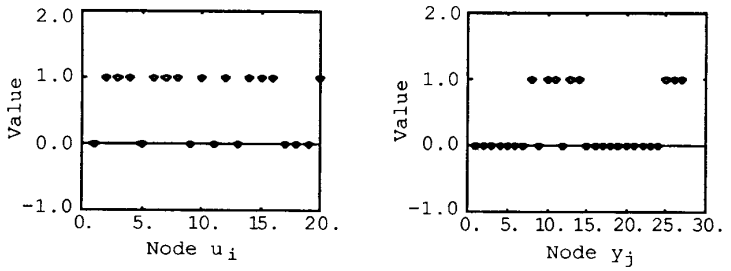

Figure 4.4 (a) The ProNet's input elements for the first WM state. (b) The ProNet's output elements for the first WM state.

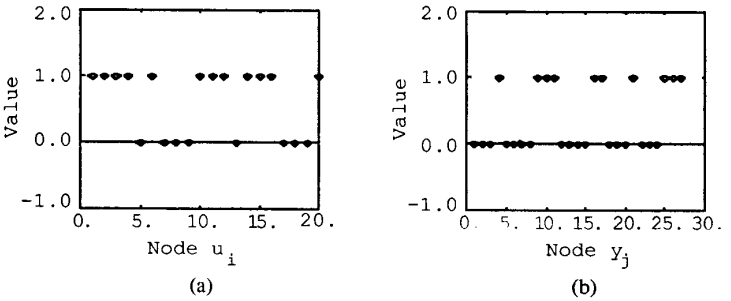

Figure 4.5 (a) The ProNet's input elements for the second WM state. (b) The ProNet's output elements for the second WM state.

Authorized licensed use limited to: UNIVERSITY NOTRE DAME. Downloaded on August 27, 2009 at 14:51 from IEEE Xplore. Restrictions apply.